Programa Educativo para Colegios Secundarios

Area Computación

Tomo 1

PROGRAMA EDUCATIVO
PARA COLEGIOS SECUNDARIOS

AREA COMPUTACION

TOMO 1

Hecho el depósito que marca la ley 11.723

© 1985 Derechos reservados por **ASWORK S.A.**
Piedras 1052 - (1070) Buenos Aires, Argentina

Prohibida la reproducción total o parcial por medios electrónicos o mecánicos, incluyendo fotocopia, grabación magnetofónica y cualquier sistema de almacenamiento de información, sin autorización escrita del Editor. Se exceptúan de esta prohibición las citas breves en artículos de periódicos o revistas destinadas a reseñar esta obra.

Primera edición, marzo de 1985

Impreso en la Argentina. Printed in Argentina

EQUIPO DE PRODUCCION

VIVIANA RUBINSTEIN
Computadora Científica (UBA)

REGINA R. DZEGHITMAN DE RUBINSTEIN
Profesora Licenciada en Ciencias de la Educación (UBA)

SUSANA SIERRA
Analista de Sistemas (UB)

JAIME CORREA
Licenciado en Educación (Universidad Católica de Chile)

ASESORES

HORACIO RODRIGUEZ
Contador Público Nacional (UBA)

PAULINA C. SOCOLOVSKY DE FRENKEL
Computadora Científica (UBA)

Realización Gráfica: **ALBERTO BENETTI**

INDICE

PROLOGO I
INTRODUCCION IV

UNIDAD 1

PRIMERA PARTE
De los dedos al ábaco 1
SEGUNDA PARTE
De John Napier (1594) a Herman Hollerith (1890) 2
TERCERA PARTE
De Howard Aiken (1937) hasta nuestros días 9
CUARTA PARTE
Aplicaciones de la computación y sus beneficios 11

UNIDAD 2

NUESTRO SISTEMA SENSORIO MOTOR 16
LA COMPUTADORA, SUS PARTES Y FUNCIONES 19
1. Unidad Central de Proceso (UCP) 20
   Aspectos electrónicos de la UCP 21
2. La Memoria 22
3. Unidad de Entrada 23
4. Unidad de Salida 23
5. Almacenamientos Auxiliares 24
EL SISTEMA BINARIO 27

UNIDAD 3

INTRODUCCION 32
1. EL PROGRAMADOR, EL PROCESADOR Y EL PROGRAMA 33
2. SISTEMA BINARIO Y LENGUAJE 34
   Lenguaje de máquina 35
   Ensamblador 35
   Lenguajes de alto nivel 35
3. EL ALGORITMO 37
EJERCICIOS 39

UNIDAD 4

INTRODUCCION 43
1. NOCIONES GENERALES DE TIMBA 45
2. ESTRUCTURA DE UN PROGRAMA EN TIMBA 45
   a) Definición del proceso algorítmico 45
   b) Descripción de los datos de las pilas 48
3. SIMBOLOGIA UTILIZADA EN LA DESCRIPCION DE SENTENCIAS 48
4. SENTENCIAS
   a) Operativas 50
   b) De selección 51
   c) De iteración 53

5. CONDICION
a) Pila vacía 55
b) Carta boca abajo 55
c) palo de carta 55
d) valor de carta 56
6. DESCRIPCION DE LOS DATOS DE LA PILA 57
7. PALABRAS RESERVADAS DE TIMBA 58
8. MENSAJE DE ERROR DE EJECUCION 59
EJERCICIOS 60

UNIDAD 5

INTRODUCCION 63
1. LA TAREA, EL INSTRUMENTO, EL PROGRAMADOR 64
2. EL PERFIL DEL PROGRAMADOR 65
3. LA PROGRAMACION ESTRUCTURADA 66
a) Desde la tarea a realizarse (problema)
hasta la instrucción ejecutable simple (sentencia) 66
b) La estructuración jerárquica 72
c) Los componentes 74
d) Estructura de datos y de programas 76
EJERCICIO 82

UNIDAD 6

INTRODUCCION 84
1. VARIABLES 85
2. ESTRUCTURAS DE DATOS 87
PILA 87
FILA 87
LISTA 87
3. EJEMPLO 88
EJERCICIOS 93

PROLOGO

Aprendizaje de la Computación y la Computación en la Educación

Marco Teórico

La Computación como elemento de la estructura Socio-Económica.

Hoy, casi en las postrimerías del siglo XX, se acepta sin discusión que la inversión destinada a la Educación debe ser rentable.

Se considera el rendimiento de una Empresa no solo en función del cálculo "costos-beneficios" sino también en función de su capacidad para desarrollarse en una estructura socioeconómica cambiante.

Los planes de desarrollo económico no solo tienen por objeto desarrollar la productividad. La esencia misma de su intencionalidad radica en elevar el nivel de vida de los individuos y de la sociedad.

Con este pensamiento se alude a la necesidad de que la institución-Escuela prepare a sus alumnos para que ellos se conviertan en personas maduras aptas para resolver sus propias vidas en la medida en que responden adecuadamente a las exigencias y cambios de la sociedad que ellos integran. Es decir, en productores y consumidores para beneficio de ellos y de la estructura en la que viven.

Así, las decisiones que las autoridades educacionales deben tomar para distribuir correctamente esa inversión, se constituyen en un desafío permanente al que deben atender analizando la estructura de la Escuela en la Argentina de hoy.

Se observa con claridad el impacto de los medios de comunicación de masas, de la actividades educativas del área Informal, de la incorporación de nuevos conceptos (Tiempo Libre, Educación Permanente, Tercera Edad, Aprendizaje Grupal), de los nuevos y poderosos alcances de la Información a partir del uso de la Electrónica y de las Computadoras, y especialmente el impacto que producen las nuevas actitudes que surgen como respuesta al avance de la Informática en diversas áreas de la sociedad y en sectores cada vez más numerosos de la ciudadanía.

En efecto, la Técnica, una vez más, produce cambios asombrosos en el campo de la Ciencia, y en el interjuego de estas dos actividades el Hombre comienza a jerarquizar los valores técnicos armoniosamente en el contexto mismo de los intelectuales y aún de los espirituales.

La informática ha contribuido al enriquecimiento del proceso de Comunicación. Enriquecimiento que modifica todas las áreas de la Sociedad (Economía, Política, Salud, Cultura, Educación...), y que se incorpora a la vida de los grupos y de las instituciones.

La Computación como motivación del Aprendizaje.

Si la Escuela Media pretende favorecer el desarrollo psicomotriz, intelectual, y afectivo de sus alumnos proporcionándoles situaciones tales que les permitan transitar experiencias de aprendizaje acordes con la realidad social, económica, cultural y política, entonces esa Escuela debe incorporar los recursos nuevos y los nuevos conceptos aplicando los métodos más adecuados de acuerdo con las teorías psicológicas y didácticas conocidas.

La Computación y las Computadoras constituyen un incentivo interesante para lograr en el alumno la motivación indispensable para todo aprendizaje, ya que los jóvenes las reconocen como instrumentos en el mundo del trabajo y del tiempo libre.

La Computación en la Escuela Media.

Es tal vez por esto que la Computación ingresa en la Escuela ante todo como recurso. Se trata de la Computación para la Educación.

La presencia de la Computadora en la Escuela satisface a los alumnos, a los padres, y a los docentes que comprenden la necesidad de abrir la Escuela a la Sociedad y sus cambios.

Pretender que la Computación integre el Plan Escolar como materia implica rescatar su escencia, definirla, e investigar de que manera su estructura lógica se relaciona con la psicológica del alumno a quien está destinada.

La ciencia de la Computación integra distintas actividades que se realizan en torno a una máquina: la Computadora, cuya característica consiste en almacenar enorme cantidad de información. Información que según instrucciones específicas, combina y sintetiza con una rapidez imposible de alcanzar por mente o grupo humano existentes.

Se pueden destacar entre las actividades de la Computación: conocer los componentes de un sistema de computación, conocer y comprender la estructura de un programa, valorar la complejidad de distintos algoritmos para resolver un mismo problema, y conocer la estructura de distintos lenguajes y sus relaciones con el lenguaje cotidiano.

La Computación y el Alumno de la Escuela Media.

El adolescente se pregunta por el sentido de sus estudios. Reflexiona acerca de si mismo y se preocupa por el presente en función de su necesidad de diseñar un proyecto de vida.

La Escuela Media reconoce esta realidad y por ello asume su función orientadora.

Advierte la necesidad de que el alumno encare el estudio a través de la inducción y la deducción. Aprenda así a observar reconociendo las variables que selecciona, a registrar los datos de esa observación y a relacionarlos sistemáticamente. Logre encontrar las semejanzas que le permitirán enunciar las hipótesis correspondientes. Sepan someterlas a prueba para verificarlas, modificarlas, o refutarlas.

Este proceso, recorrido en niveles de abstracción creciente, permitirá el abordaje, en los últimos años de la Escuela Media, del ejercicio y la reflexión acerca de sus propias deducciones.

Esta afirmación fundamenta la propuesta de incorporar en ese nivel la enseñanza de la Computación, ya que ella, por su naturaleza, proporciona la posibilidad de aplicar el pensamiento analítico, y el procesamiento de la información en alto nivel de abstracción.

Contribuye de este modo el aprendizaje de la Computación a estructurar el pensamiento lógico del alumno, su capacidad de autoevaluación, y su autodisciplina.

Ayuda al alumno a desarrollar sus hábitos, destrezas, y habilidades.

Aprender Computación significa enfrentarse con la posibilidad de conocer las estructuras de una máquina y la de los programas y sistemas operativos.

Comprender la interacción que existe entre ellas.

Es decir, instrumentarse para observar el mundo circundante con actitud científica. Con método científico.

Analizar cuidadosamente las partes de un programa para ponerlo a prueba, esto es, ejercitar el análisis y aplicar el método para construir programas, y principalmente aprender la metodología para resolver problemas, son los logros más importantes que brinda el conocimiento de la Computación a sus alumnos.

De esta manera contribuye también a que ellos accedan a la comprensión de la problemática del conocimiento.

Que ellos integren en un todo estructurado la Lógica subyacente en todo proceso de aprendizaje.

Que ellos valoren la interrelación que existe entre Lenguaje y Pensamiento.

La Computación y los Docentes de la Escuela Media.

Instalar un Laboratorio de Computación en la Escuela Media no solo significa introducir máquinas y terminales, sino también organizar un grupo de profesionales y docentes interactuando con la institución y sus alumnos.

Instrumentar, con docentes en Computación, equipos de profesores interesados en usar la Computación como recurso, y deseosos de conocer las reales posibilidades de esta ciencia.

Brindarles programas acordes con la naturaleza de su asignatura.

En fin, en torno al Laboratorio de Computación organizar actividades integradoras coordinadas con criterio didáctico.;

Se pretende de esta manera lograr una contribución al enfoque interdisciplinario, y promover una mayor comunicación entre los docentes de las distintas asignaturas.

Finalmente se aspira a organizar un Laboratorio de Computación para lograr un refuerzo de la integración Docentes-Alumnos, y del proceso de Enseñanza y su correlato: el de Aprendizaje.

Introducción

El juego del "gallo ciego" nos puede servir para explicar lo que pretendemos. Nos dirán que ese juego es cosa de chicos, algo que dejaron atrás en su infancia. Pero muchos estudiantes secundarios terminan el colegio para jugar al "gallo ciego", para elegir cualquier cosa.

En realidad, no para elegir, sino para tomar lo primero que tocan con los dedos. Muchos salen de la Universidad para jugar al "gallo ciego". Y muchos se pasan la vida en esto.

El estudiante secundario se está preparando para ocupar un nuevo lugar en esta sociedad. Un lugar más activo, más influyente. Ya sea para transformarla, ya para conservarla, tienen que decidirse para ocupar un nuevo lugar en esta sociedad.

La computación es un elemento importante para comprender el mundo en el cual viven. El agua, el gas, la luz —tres elementos vitales de la vida cotidiana— son pagados mediante sistemas de computación y los ejemplos podrían multiplicarse en páginas y páginas.

Un buen libro, una buena película, una buena charla... les va quitando la venda de los ojos. Cada cosa que aprenden les permiten ver más y mejor el futuro, el mañana.

Con este curso pretendemos: que comprendan mejor el mundo en el cual viven y su mismo funcionamiento. Y que se quiten un poco la venda de los ojos.

Por supuesto, como todas las cosas, la Computación es el fruto del trabajo de miles de hombres, a lo largo de miles de años, hasta nuestro días. No es cosa de un pueblo genial, sino de seres humanos que, en el tiempo, fueron dando pequeños y grandes pasos, pero siempre apoyándose unos sobre otros.

Actualmente, la computación se usa para las armas de guerra, para el desarrollo económico, o para el acceso de más personas a la cultura y al saber.

Es un instrumento más, para usarlo en un mundo, donde hay mucho, muchísimo por hacer.

Qué es la Computación?
Para qué aprender Computación?
Porqué?.....

Trataremos de guiarlos a través del curso para que den respuesta a estos interrogantes. A estos problemas.

Decidimos comenzar abordando algunos jalones del camino trazado por el Hombre que condujeron a la situación actual de la Computación con el propósito de que los integrantes de los grupos de estudio después de la lectura y el análisis del texto de esta Unidad, y la posterior discusión del grupo y su coordinador, logren acceder a los siguientes:

• Objetivos Generales

1. Adquieran información acerca de la continuidad de los esfuerzos del Hombre para interactuar adecuadamente con su medio.
2. Comprendan que la Ciencia y la Técnica constituyen una unidad dinámica
3. Comprendan que el Hombre siempre ha creado Instrumentos para resolver sus Problemas.
4. Valoren su rol en el proceso de su propio aprendizaje.

• Objetivos Específicos

1.1. Describan situaciones correspondientes a dos épocas históricas distintas y sus consecuencias en el área técnica.

1.2. Enumeren tres factores que posibilitaron a Blas Pascal la creación de su Máquina.

1.3. Definan la obra de Joseph Marie Jacquard en base a sus características más destacadas.

1.4. Enuncien actividades que puedan realizarse con las computadoras

1.5. Enuncien actividades que no puedan realizarse con las computadoras.

2.1. Analicen las posibilidades y limitaciones del pensamiento del Hombre que usa el Abaco y las del que usa los Huesos de Napier.

2.2. Señalen las semejanzas en el proceso del pensar en el Hombre que usa los Dedos para Contar y en el que usa el Contador de Arena.

2.3. Expliquen las razones por las cuales la Máquina Analítica de Babbage constituye una etapa fundamental en la Historia de la Computación.

2.4. Relacionen en la siguientes situaciones Pensamiento e Instrumento
— Contar con los Dedos.
— Usar la Edvac de Von Neumann.

2.5. Expliquen las razones por las cuales la máquina de John Van Neumann significa un aporte en la Historia de la Computadora.

3.1. Analicen las situaciones que llevaron a Jacquard y a Pascal a la construcción de sus máquinas.

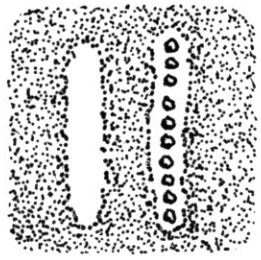
3.2. Relacionen la preparación profesional de Hollerit, su capacidad para recrear Técnicas anteriores, y para utilizar nuevas formas de energía, con sus aportes en el campo de la Computación.

4.1. Participen activamente en las reuniones del grupo de aprendizaje formulando preguntas significativas.
4.2. Exprésen su interés cada vez que lo perciban.
4.3. Juzguen la unidad del Oriente y del Occidente en la Historia de la Ciencia y de la Técnica.
4.4. Expresen su conformidad por aquellos aspectos de la Planificación, Conducción y Evaluación cada vez que los consideren necesario.
4.5. Aporten otros datos para reforzar este enfoque.
4.6. Intenten autoevaluarse en un análisis grupal de su desempeño.
4.7. Asuman el rol protagónico en su propio aprendizaje.

Primera Parte: De los dedos al ábaco.

Primitivamente las cuentas se hacían con los dedos. Nos referimos a más de 5.000 años de nuestra era actual.

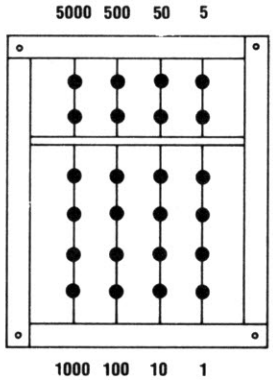
Con los dedos se comunicaban los números, y mediante ciertas posiciones se realizaban adiciones y sustracciones sencillas, aumentando o disminuyendo el número de dedos desplazados.



Pero con el desarrollo del comercio, cuando los mercaderes viajaban de un lado para el otro, comprando, vendiendo y haciendo trueques, sus dedos no alcanzaban para obtener toda la información de su intercambio comercial.

Fue entonces cuando los egipcios, inventaron el "contador de arena", uno de los primeros medios mecánicos para calcular.

Se trazaban surcos verticales en la arena, se tomaban piedritas y se establecía una correspondencia entre los objetos a contar y las piedras. Al completar 10 piedritas en el surco de las unidades, se cambiaban por una sola en el surco siguiente de la izquierda.



Pero el instrumento que revolucionó la computación en aquella época (unos 4.000 años atrás) fue el **ábaco**, inventado y construído en Oriente, probablemente en China.

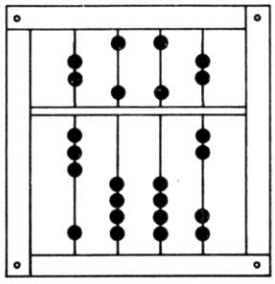
Una fila de varillas verticales y paralelas entre sí, y donde se deslizan pequeñas bolitas, están montadas sobre un armazón rectangular.

Las bolitas se dividen en dos secciones, por un travesaño que las separa.

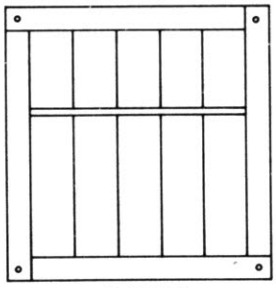
Las bolitas que están bajo el travesaño representan los siguientes valores: las encolumnadas en la primera varilla de la derecha, representan 1 cada una; las de la segunda de la derecha, 10 cada una; las de la tercera, 100; las de la cuarta, 1.000 cada una.

Las bolitas que están sobre el travesaño, representan: las encolumnadas en la primera varilla de la derecha, 5 cada una; las bolitas de la segunda varilla, 50; las de la tercera, 500; y las de la cuarta, 5.000 cada una.

La entrada de los números se realiza moviendo las bolitas hacia arriba (o hacia adelante, según la posición del ábaco).



CUADRO A.



CUADRO B

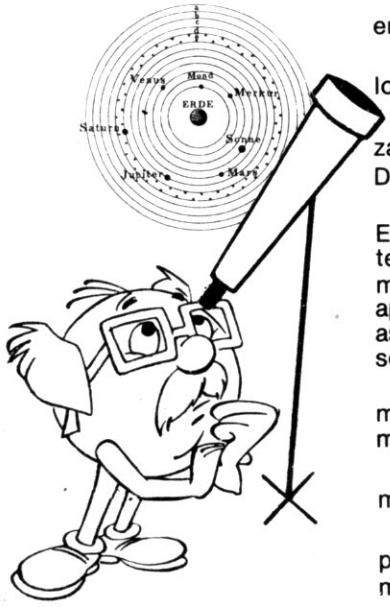
Si queremos, por ejemplo, representar la cifra 3552, adelantamos las bolitas correspondientes, y la lectura del ábaco será como se ve en el cuadro "A".

Ahora llenen Uds. mismos con el lápiz, (cuadro B) las bolitas que habría que adelantar para escribir la cifra 12.342, pero, a fin de que se familiaricen con el ábaco, comiencen desde la derecha: primero ubiquen el 2, después el 40, luego el 300, el 2.000 y el 10.000.

Aprendieron a entrar números y a leer en el ábaco. Escribieron una cifra que, con el contador de arena (ver pág. 1) les hubiera requerido una tarea muy esforzada, llena de riesgos de error y poco práctica. Esto nos da una idea del salto fabuloso que significó este instrumento, pequeño, portátil y capaz de almacenar grandes cifras. Pero no sólo de almacenar, sino también de sumar, restar, multiplicar y dividir, todo ello a una velocidad asombrosa. Las sumas (adiciones) se hacen adelantando bolitas, las restas (sustracciones) invirtiendo el proceso de la adición. La multiplicación y la división se hacen con sucesivas adiciones y sustracciones.

Los ábacos fueron los aparatos de cálculo más importantes durante más de 4.000 años y aún se usan en el Oriente.

Segunda Parte: De John Napier (1594) a Herman Hollerith (1890).



Los próximos instrumentos de cálculo, aparecen en Europa, en el Siglo XVI.

Los aportes que Occidente hace en esta materia, se deben a los conocimientos que antes asimiló del Oriente.

Desde luego, el ábaco se impuso entre los romanos, desplazando a sus "calculi", unas piedrecitas que utilizaban para contar. De ahí el origen de nuestra palabra "cálculo".

En el transcurso de la Edad Media, el ábaco se impuso en Europa. Pero en este período histórico, se da un proceso importante: los **árabes** transmiten desde España, sus conocimientos matemáticos basados en el **sistema decimal**, conocimiento que a su vez aprendieron de la India. También transmiten sus conocimientos de astronomía aprendidos de los chinos, y los conocimientos de filosofía, aprendidos de los griegos.

A través de un largo proceso, los europeos asimilan el sistema indo-arábigo de matemáticas, y también desarrollan la astronomía, una actividad que requiere mucho cálculo.

A fines del Siglo XVI aparece **John Napier**, escocés, un ilustre matemático dedicado a la astronomía, entre otras materias.

Napier sistematizó el uso del **punto decimal**, para separar la parte fraccionaria de la parte entera de los números. En 1594 comenzó a trabajar con **logaritmos**, a él debemos su invento.

Fig. 1 Fig. 2 Fig. 3

En materia de instrumentos de cálculo, el gran aporte de Napier son sus famosas y populares "**varillas**", o también llamados "**huesos de Napier**". El asignó una varilla a cada número, del 0 al 9, y escribió los múltiplos de cada uno en cuadraditos.

Por ejemplo, la varilla del 4. Debajo suyo están: 8, 12, 16, etc. (Fig. 1).

Pero Napier separó la unidad de las decenas, de cada múltiplo, mediante una diagonal en cada cuadradito. (Fig. 2).

Llenen la varilla del 3 según el sistema de Napier. (Fig. 3).

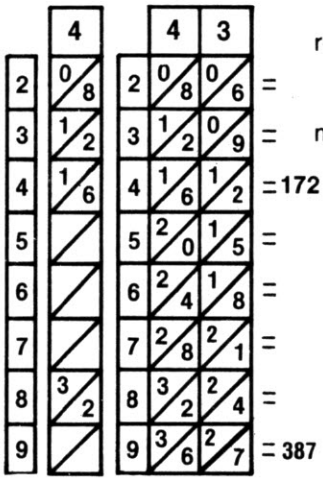


Fig. 4 Fig. 5

Si colocamos junto a la varilla del 4, otra varilla con los números corridos del 2 al 9, tenemos la tabla de multiplicar del 4. (Fig. 4).

Completen la figura N° 4.

Si hacemos lo mismo con las varillas del 4 y del 3, podemos multiplicar, mediante simple suma, por 43. (Fig. 5).

Multipliquemos 9 × 43.

En las varillas, tenemos el valor de 9 × 4 y el valor de 9 × 3.

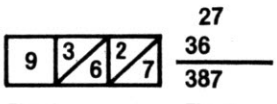
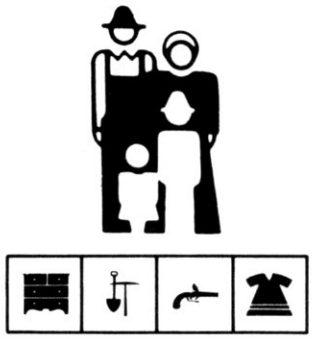


Fig. 6 Fig. 7

En la Fig. 6 hemos aislado la lectura del 9 × 43. Ahora sumamos como en la Fig. 7 y tenemos el resultado de 9 × 43.

Completen Uds. la tabla del 43. (fig. 5)



Los grandes descubrimientos geográficos del Siglo XVI, van permitiendo a Europa transitar, desde una economía de autosubsistencia, a otra de mercado. O sea, se comienza a producir, no ya para satisfacer las necesidades de una comunidad, sino también para intercambiar productos propios con los de otras comunidades. Esto permitió que las economías europeas desarrollasen su artesanía (producción de muebles, relojes, herramientas de todo tipo, armas, ropa, etc), una actividad de tipo familiar que se ubica en las ciudades, que por entonces se llamaban "burgos".

Los "burgos" europeos comienzan a crecer, son asiento de actividades artesanales, comerciales, bancarias, políticas y administrativas, además de universitarias.

Blas Pascal nace en uno de esos grandes burgos de la Europa de comienzos del Siglo XVII: París. Su padre era un típico burgués ("hombre de los burgos"), perteneciente al régimen administrativo del reino, como recaudador de impuestos.

Esta era una tarea que se hacía cada vez más compleja, en una sociedad que también se tornaba cada vez más compleja.

Pascal revolucionó su época por sus aportes en el campo de la filosofía, pero también fue un genio matemático, y además revolucionó el campo de la computación.

A fin de aliviar las tareas del nègocio de contabilidad de su padre, y aprovechando el desarrollo de la artesanía metalúrgica, que ya construía piezas con gran precisión, ideó y construyó un instrumento mecánico para sumar y restar.

Fue en el año 1644.

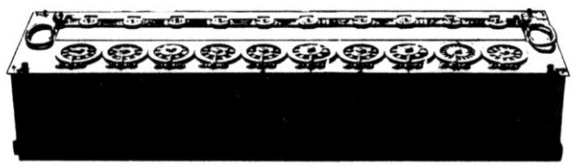
Esta máquina consistía en una serie de ruedas cuyas posiciones podían ser observadas a través de un visor en la tapa de la caja que encerraba el mecanismo.

Los números se entraban por medio de un disco similar al del teléfono.

En una serie de diez ruedas que tenían dientes para representar los números, estaban grabados los dígitos del cero al nueve. La rueda de la derecha representaba las unidades, la segunda las decenas, la tercera las centenas, etc, etc,

Mediante un juego de engranajes, la próxima rueda de la izquierda giraba una unidad cuando la anterior excedía el nueve.

La máquina de Pascal era un dispositivo digital, que realizaba los cálculos según las reglas de la aritmética entera.

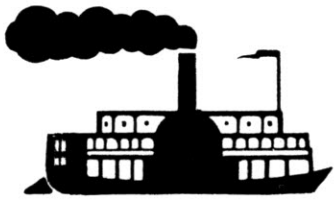


Años más tarde, alrededor de 1666, **Sir Samuel Moreland**, perfeccionó la máquina de Pascal, logrando la multiplicación mediante técnicas de sumas acumulativas.

Hacia fines del Siglo XVII, en 1694, el alemán **Gottfried W. Leibnitz** concluye su "Contador Escalonado", una máquina más avanzada que la de Pascal, capaz de sumar, restar, multiplicar, dividir y extraer raíces cuadradas. El proceso se realizaba por una serie de sumas repetidas, al igual que las actuales computadoras digitales.

La mayor contribución de Leibnitz fue demostrar las ventajas del **sistema binario** sobre el decimal.

El Siglo XVIII no presenta avances notorios en el terreno de la computación. Pero éste es un siglo muy rico en hechos históricos, que prepara futuros avances.



En el campo político, Europa reemplaza los sistemas monárquicos de gobierno por los democráticos; y en el campo económico reemplaza los sistemas **artesanales** de producción, por los industriales.

La **revolución industrial** se despliega en el Siglo XVIII con la introducción de una nueva fuente de energía: el vapor. El escocés Santiago Watt (nacido en 1736) concibe el principio de la máquina a vapor de doble efecto, que se aplicará para mover máquinas, barcos, carros y locomotoras.

En este marco de avances tecnológicos, un ingenioso francés inventa un telar automático, que tendrá enormes repercusiones en el área de la computación, incluso hasta nuestros días.

En efecto, **Joseph Marie Jacquard** era un ingeniero de la industria textil, que no pensaba en como mejorar las máquinas calculadoras, sino la producción de ropa.

Preparó una cadena sin fin de tarjetas perforadas, para que girara pasando por las agujas de un telar especialmente diseñado. Sólo cuando una aguja coincide con uno de los agujeros de la tarjeta, puede penetrar, e incorporar un hilo en el tejido. Cada cadena de tarjetas constituiría un verdadero **programa** para la máquina tejedora, la cual podía incorporar sobre un paño, toda suerte de figuras y diseños, según la ubicación de los agujeros en las tarjetas.

El telar automático de Jacquard se incorporó a la producción a comienzos de 1800, aunque su primer modelo fue concluído varios años antes.

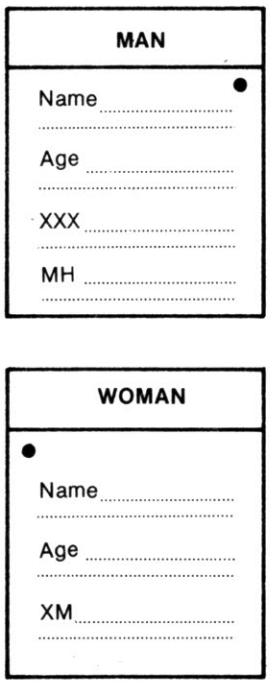
En el año 1835, el matemático inglés **Charles Babbage**, concibe su "máquina analítica", primera computadora de la historia, utilizando la tecnología de la tejedora de Jacquard. La **máquina analítica** utilizaba dos conjuntos de tarjetas perforadas de Jacquard: uno con datos codificados que constituía una unidad de **almacenamiento**; el otro conjunto de tarjetas perforadas tenía una secuencia de operaciones, constituyendo una unidad de **procesamiento** de los datos.

Los émbolos pasaban a través de las perforaciones de las tarjetas y operaban los mecanismos para transferir los números del almacenamiento a la unidad de procesamiento de datos.

Así como el telar automático de Jacquard tejía flores y hojas, la Máquina Analítica de Babbage **"tejía"** modelos algebráicos, de un modo también autocontrolado.

Lo más notable de la **Máquina Analítica** es que contenía todos los rasgos esenciales que forman la computadora electrónica moderna.

Sin embargo, Charles Babbage fue un genio adelantado a su época. Preparó planes detallados para la concreción de su máquina, pero no existían las condiciones tecnológicas para consumarlos. Sus escritos se sumergieron en el olvido y un siglo después, en 1937, son redescubiertos, para servir de apoyo a nuevos adelantos.



La tecnología de los telares de Jacquard es retomada por otro pionero de la computación, el estadístico norteamericano **Herman Hollerith**.

Hollerith trabajó en el censo de los Estados Unidos de 1890. Diez años antes, se habían censado todos los habitantes de este país, y los administradores de la Oficina de Censos habían llegado a la conclusión de que, cuando las cifras estaban por fin clasificadas, no eran más utilizables. En 1887, ya próximos a la fecha del nuevo censo, se comenzaron a buscar procedimientos que permitieran manejar con mayor eficiencia, una masa de datos que iba a ser mucho mayor.

Entre otras alternativas, algunos propusieron reemplazar el viejo sistema de los libros, por tarjetas de colores y signos para facilitar el cálculo.

Hollerith recurrió a las tarjetas perforadas de Jacquard. Asignó una ficha a cada persona censada y perforó, a la derecha, las de los hombres y a la izquierda, las de las mujeres.

La perforación de la ficha, permite que una determinada varilla metálica se introduzca en un dispositivo que acciona un contador. Cada uno de los dos contadores —el de los hombres o el de las mujeres— va sumando las tarjetas correspondientes y almacenando las cifras.

Estos contadores fueron construídos según la tecnología ideada por Blas Pascal, para su máquina calculadora.

Luego Hollerith dio un paso más, utilizando la energía eléctrica para accionar los contadores. En lugar de varillas metálicas, por el agujero de la ficha pasa la electricidad, hacia uno de los dos contadores: el de los hombres, o el de las mujeres.

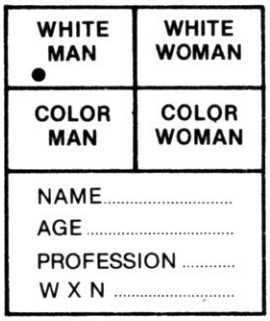
Pero el censo de los Estados Unidos discriminaba además de los hombres y las mujeres, a los blancos de los negros.

Hollerith diseñó una nueva tarjeta con cuatro casilleros: para los hombres blancos, los hombres negros, las mujeres blancas y las mujeres negras.

Según de quien se tratase, se le asignaba una tarjeta perforada en uno de los cuatro casilleros.

Y conectó cuatro contadores en su máquina.

El censo de 1890 ocupó la tercera parte del tiempo del de 1880, no obstante que, en los 10 años transcurridos, la población se incrementó considerablemente.



A una velocidad de "lectura" de 80 tarjetas por minuto, la máquina de Hollerith computó y clasificó, en 6 días, a 62.622.250 habitantes.

Resumiendo, H. Hollerith integró dos tecnologías básicas: la de Jacquard y la de Pascal. La primera, reemplaza al hombre en acciones concretas, como tejer; la segunda, en cálculos mentales, como sumar. Esta nueva máquina, que además incorporó la energía eléctrica para su movimiento, reemplazó al hombre en la pesada y tediosa tarea de **leer** millones de fichas para sumarlas clasificadamente. A partir de una masa desordenada de **datos** produjo una **información**, todo ello a gran velocidad.

Hollerith, incorporando elementos ya inventados, los aplica a un **uso** hasta entonces desconocido: **la manipulación de grandes volúmenes de datos**.

Herman Hollerith fue uno de los socios fundadores de la International Bussiness Machine (I.B.M.).

Cuadro sinóptico hasta el siglo XIX			
FECHA	**LUGAR**	**INVENTOR**	**FUNCIONES Y CARACTERISTICAS**
Más de 5000 años atrás			Uso de los dedos, para sumar o restar.
Más de 4000 años atrás	Medio Oriente	¿Egipcios?	Contador de arena Permite sumar o restar y agrupar de a 10 unidades.
4000 años atrás	Lejano Oriente	¿Chinos?	Abaco. Permite sumar, restar, multiplicar y dividir.
Siglo XVI	Escocia	John Napier	Varillas o huesecillos. Basado en el sistema decimal y uso de Número Arábigo. Permite multiplicar con varios dígitos.
Siglo XVII	Francia	Blas Pascal	Máquina de sumar y restar. Primer sistema mecánico.
Siglo XVII	Inglaterra	Samuel Moreland	Perfecciona a Pascal. Su máquina suma, resta y multiplica.
Siglo XVII	Alemania	G. Leibnitz	Contador escalonado. Puede sumar, restar, multiplicar, dividir y extraer raíces cuadradas. Basado en sistema binario.
Siglo XVIII	Francia	J.M. Jacquard	Aporta involuntariamente a la computación con su telar automático, a base de tarjetas perforadas.
Siglo XIX	Inglaterra	C. Babbage	Máquina analítica. Utiliza tarjetas perforadas. Contiene todas las características de las computadoras electrónicas. No se construye.
Siglo XIX	EE.UU.	H. Hollerith	Combinación de la técnica de Jacquard y la de Pascal. Incorpora electricidad. Lee, cuenta y clasifica tarjetas del censo.

Tercera Parte: De Howard Aiken (1937) hasta nuestros días.

Los próximos avances en materia de computación, se registran en la década que va de 1930 a 1940.

Tengamos presente que es ésta una década bastante convulsionada, que se inicia con una crisis económica mundial, la que a su vez prepara el estallido de la Segunda Guerra Mundial en 1938, un conflicto que involucra directamente a todas las potencias industriales del mundo.

Estados Unidos se mantiene al margen de esta guerra, que tiene por escenario a Europa, para ingresar en ella recién en 1942.

Mientras Europa está completamente absorbida por esta guerra, Estados Unidos de Norteamérica inaugura su actual liderazgo industrial y militar en el mundo.

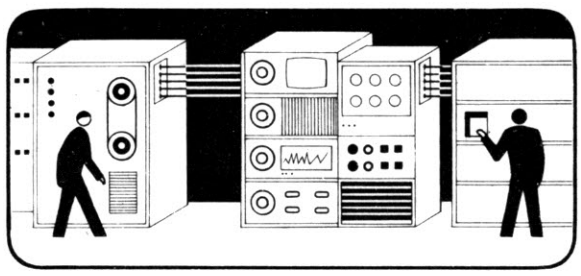
Esta situación internacional nos explica, por una parte, que los próximos adelantos en materia de computación proceden, básicamente, de los Estados Unidos; y por la otra, que las aplicaciones de estos nuevos instrumentos se realicen asiduamente en el campo militar: la balística, la cohetería y la energía nuclear.

Howard Aiken, profesor de la Universidad de Harvard, fue el primero en examinar y estudiar los escritos olvidados de Charles Babbage. Era el año 1937.

Ahora, con los avances tecnológicos, especialmente la electricidad, se podían poner en práctica las funciones básicas de la Máquina Analítica de Babbage.

La idea de Aiken apuntaba hacia la obtención de una máquina calculadora **totalmente automática**. En 1939 se asoció a los ingenerios de I.B.M. y trabajó durante 5 años en la construcción de la MARK I, una calculadora automática, **electromecánica**, de aproximadamente 15 metros de largo por 2,40 metros de alto. Era accionada y controlada por una cinta de papel perforada. Todos los números eran representados por 23 dígitos decimales y realizaba 5 operaciones fundamentales: adición, sustracción, multiplicación, división y referencia a tablas. La entrada era por medio de tarjetas perforadas y las salidas se registraban, tanto en tarjetas perforadas como en máquinas de escribir eléctricas. Al igual que sus predecesores, desde Pascal hasta Hollerith, utilizó registros mecánicos para almacenar números.

La segunda calculadora del profesor Aiken, la MARK II, era más rápida que la primera, gracias a que reemplazó el sistema electromecánico por otro, de tipo **electrómagnético**. La MARK II aparece hacia fines de la guerra mundial, y su utilización fue militar: cálculo de funciones trigonométricas, de objetivos ópticos y de trayectorias de cohetes.



J. Presper Eckert y **John W. Mauchly**; ambos de la Escuela Electrónica Moore, de la Universidad de Pennsylvania, inventaron y terminaron de construir en 1946 la ENIAC (calculador e integrador numérico electrónico), la primera computadora realmente **electrónica**. Esta máquina ya no necesita ser controlada externamente (por tarjetas perforadas o por cintas de papel perforadas) sino a través de cables integrados a su propio cuerpo.

La ENIAC era cien veces más rápida que todas las máquinas electromecánicas de entonces. Podía realizar 5000 cálculos por segundo y necesitaba un espacio de 150 m^2. Fue utilizada por los físicos nucleares, y se estimó que los cálculos que éstos realizaban durante dos horas con ella, hubiera demandado un año de trabajo a cien ingenieros.

En 1950, Eckert y Mauchly organizan su propia compañía: la Electronic Control Company.

En 1947, **John Von Neumann** ideó un nuevo concepto: una máquina de **programa almacenado**, que reemplaza a las máquinas externamente programadas, hasta entonces.

Este nuevo concepto de Von Neumann fue llevado a la práctica por Eckert y Mauchly, quienes construyeron la EDVAC, concluida en 1952. Tal como había sido sugerido por Babbage, la EDVAC fue capaz de operar y cambiar las instrucciones almacenadas.

La EDVAC, más poderosa que la ENIAC, utilizaba la notación del sistema numérico binario y estaba totalmente programada internamente.

En la década del 50 se inicia la etapa de la fabricación de computadoras para ser ofrecidas al mercado.

Esta etapa se inicia con la UNIVAC I (computadora automática universal), fabricada en 1951 por la Remington Rand Corporation, que fue utilizada por la Oficina de Censos de los Estados Unidos primero y para el procesamiento de datos comerciales, después.

La segunda computadora comercial fue la CRC 102, fabricada en 1952 por la Computer Research Corporation (NCR 102).

También en 1952 I.B.M. sacó al mercado la I.B.M. 701. Luego, en 1953, I.B.M. 702 e I.B.M. 650

La producción, comercialización y uso de las computadoras recibe un impulso decisivo con el invento del **transistor**, ocurrido en 1948 y aplicado a las máquinas calculadoras a partir de 1959.

La computadora transistorizada es más pequeña y más confiable que las grandes moles hechas con tubos al vacío.

A comienzos de 1960, los fabricantes de transistores lograron colocar circuitos electrónicos completos sobre la superficie de una pequeña oblea de sílice (chips), llamados **circuitos integrados**.

En 1970, I.B.M. sacó al mercado una computadora de circuito integrado (sistema 370).

En la actualidad, el gran avance es la **micro-computadora**, producída desde muchas empresas y masivamente comercializada en todo el mundo.

En noviembre de 1970 la Intel Corporation da un paso decisivo en la historia de la electrónica al fabricar el primer **microprocesador**, es decir, la computadora sobre un **"chip"**. Este pequeño dispositivo electrónico, había reemplazado la totalidad de los 18.800 tubos al vacío de la ENIAC.

Gracias al tamaño y peso logrado, por el descubrimiento del transistor, hoy existen computadoras que controlan lavarropas, despertadores, televisores, etc..

El límite de su uso está en nuestra imaginación, no en la tecnología.

Cuarta Parte: Aplicaciones de la computación y sus beneficios.

La computación ha producido grandes transformaciones en nuestra época y las continuará produciendo.

En el comercio, en la producción, en la política, en la investigación científica, en la guerra... en cualquier actividad que pensemos, **el manejo de la información** da ventajas.

No basta que una decisión sea correcta, debe ser también oportuna, aplicada en el momento preciso. Una información oportuna, permite tomar decisiones oportunas.

Hoy día se producen grandes cantidades de datos, que deben ser almacenados y procesados con rapidez.

Además, con el desarrollo de las telecomunicaciones, podemos anular las distancias y conocer al instante lo que ocurre en los mercados, en los gobiernos o en las fábricas de cualquier lugar del mundo, (alzas o bajas de monedas, decretos y leyes que afecten el intercambio mundial, lanzamiento de nuevos productos, etc).

La computación, entonces, no se limita al cálculo rápido, nos permite también manejar información.

A continuación, se resumen las aplicaciones más comunes de la computación en diversas actividades y tareas.

—Reemplaza al hombre en tareas rutinarias, aburridas, y faltas de creatividad.

—Se puede aplicar a todo lo que es producción en serie: montar, unir, atornillar, etc.

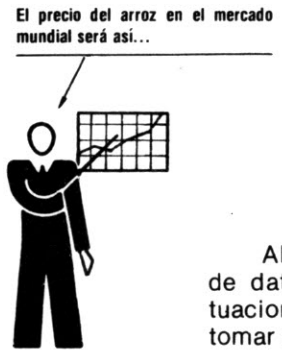
—Puede ser beneficioso según las circunstancias (problema de desempleo).

—Auxilia en problemas relativos a incremento de volumen.

Por Ej.: el fabricante desea saber, con rapidez y exactitud, si puede venderle a un exportador ocasional, sin desatender a los clientes que le compran todo el año, cantidades más o menos constantes.

—Libera a muchas personas de tareas administrativas y contables: facturaciones, remitos, impuestos, descuentos, etc. etc.

Estas personas pueden dedicarse a tareas más creativas y que la computadora no puede hacer: búsqueda de objetivos, establecimiento de criterios, etc.



Al procesar grandes cantidades de datos, podemos pronosticar situaciones futuras. Y esto nos permite tomar decisiones más acertadas.

En síntesis, la computadora reemplaza al hombre en tareas de tipo rutinario, en cálculos agobiantes y complicados, y permite manejar importante cantidad de información en brevísimo tiempo.

En las actividades que requieren creatividad, en la definición de criterios de acción y en tomas de decisión por voluntad, gusto o inspiración, las computadoras no nos pueden reemplazar. Nos ayudan indirectamente, porque nos dejan más tiempo disponible para ésto.

Qué es la Computación?

Para qué aprender Computación?

Porqué?.....

Prometimos guiarlos a través del curso para que den respuesta a estos interrogantes. A estos problemas.

Comenzaron estudiando la Historia de la Computación, y han intentado acceder al logro de los objetivos enunciados en la Unidad I.

Les proponemos ahora la lectura y el análisis del texto "La Máquina" y su posterior discusión en grupo, con la conducción de su coordinador.

Cuando cumplan con estas actividades lean nuevamente el enunciado de cada uno de los objetivos generales de la Unidad I y verifiquen si han reforzado esos logros.

Realicen este ejercicio cada vez que estudien una Unidad. Así podrán evaluar si deben revisar la anterior, o si pueden seguir adelante con éxito

Para evaluar el trabajo realizado con el tema "La Máquina" hemos seleccionado y formulado, en términos de los integrantes de los grupos de aprendizaje los siguientes

• Objetivos Generales

1. Adquieran información acerca de la estructura física de la Máquina.
2. Adquieran información acerca de las funciones de cada parte.
3. Comprendan la relación que existe entre la Computadora y nuestras funciones cerebrales, sensitivas, motoras.
4. Valoren la importancia de la información para acceder a la comprensión.

• Objetivos Específicos

1.1. Enumeren las partes de una Computadora.
1.2. Describan la parte externa de la Máquina.
1.3. Asignen nombre a los componentes de la parte interna.
2.1. Indiquen los elementos que ingresan por la parte externa.
2.2. Informen acerca de la función del dispositivo de salida de la parte externa.
2.3. Mencionen los elementos que se almacenan en la Memoria de la Computadora.
2.4. Realicen un diagrama de la parte interna de la Computadora para señalar las funciones de la Unidad de Control y las de la Unidad Aritmético-Lógica.
3.1. Enuncien funciones del Cerebro Humano que se asemejan a las de la Memoria, Unidad Aritmético-Lógica y Unidad de Control de la Computadora.
4.1. Participen activamente para adquirir información acerca de los aspectos electrónicos de la Microcomputadora.

4.2. Se esfuercen por aprender la estructura del Sistema Binario.
4.3. Contribuyan a esclarecer las relaciones que existen entre la estructura de la Máquina y su uso.

Nuestro Sistema Sensorio Motor.

La computadora representa una página más en la historia de los instrumentos creados por el hombre.

En efecto, la historia del hombre se confunde con la historia de sus propios instrumentos. El más perfecto de todos ellos es, sin duda, la mano.

El "efecto pinza", que consiste en apretar la punta del pulgar contra la punta de los cuatro dedos restantes, es una capacidad exclusiva del hombre.

Este, educando sus manos, produce el gran instrumento natural, que es fuente de cuanto instrumento ha sido creado en esta tierra.

Por cada instrumento, el hombre aumenta su capacidad de hacer algo: con la pinza aumenta su capacidad de apretar; con el teléfono, su capacidad de escuchar y hablar; con el auto su capacidad de correr...

El hombre construyó la computadora para aumentar, en primer lugar, su capacidad de acumular y procesar información, resolver problemas y realizar complicados cálculos matemáticos, todo ello en brevísimo tiempo y con un mínimo esfuerzo.

Veamos esto más detenidamente.

El hombre hace cosas a su imagen y semejanza, y también a imagen y semejanza de otros seres vivos.

El computador tiene algunas semejanzas con el mundo animal, incluyendo a los seres humanos.

Estamos conectados con el mundo a través de nuestros sentidos, que captan diversos DATOS que ocurren a nuestro alrededor.

Nuestros sentidos forman un sistema de ENTRADA, por ellos entran ruidos, imágenes, olores y otras sensaciones.

Los datos que entran en nosotros, producen diversas reacciones. Un bocinazo en la calle nos hace detenernos y girar la cabeza hacia el coche; mientras que una música de zamba, nos hace, por ejemplo, golpear el piso con la suela del zapato.

Estas conductas (detenerse y girar la cabeza, zapatear...) SALEN de nosotros, son observables por los demás.

Nuestras conductas son SALIDAS y nuestros miembros son un sistema de salida.

La gente que nos observa puede "leer", digamos, las acciones que sacamos afuera.

Según el dato que entra, producimos una determinada salida.

Pero nadie zapatea zamba ante un bocinazo, ni se detiene bruscamente y gira, ante el ritmo de una zamba...

Esto quiere decir que, entre el dato que **entra** y la conducta que **sale**, hay un PROCESAMIENTO. El dato es procesado, es sometido a una cierta evaluación e interpretación, antes de actuar.

Los datos que "entran" en nosotros, además de procesados, son ALMACENADOS, son guardados. Ese bocinazo en la calle se ALMACENA, digamos, en nuestra memoria. Gracias a este almacenamiento es que podemos procesar. Otro bocinazo más, será comparado con el bocinazo anterior que se almacenó, y nos indicará que estamos frente a una nueva situación de peligro y nos detenemos. Otro individuo que nunca estuvo en la ciudad echará a correr, y será quizás atropellado.

Es que, al no almacenar experiencias anteriores, no supo PROCESAR correctamente.

Una cosa inversa nos ocurrirá a nosotros en la selva, por ejemplo, porque de allí, no tenemos experiencias almacenadas.

A nuestras conductas las hemos llamado "salidas". Ellas son acciones que proceden de un centro de CONTROL, que ordena una determinada conducta. Zapateamos y movemos el cuerpo, porque a éste se lo hemos ordenado desde nuestro cerebro, adonde "entró" la música y se procesó.

Cada movimiento que hacemos pasa por un centro de control.

Así como la pinza o la llave inglesa, se parecen a nuestros dedos pulgar e índice y en cierto modo funcionan como nuestras manos, del mismo modo un computador funciona como nuestro SISTEMA SENSORIO MOTOR, y el sistema sensorio motor de todos los animales.

En resumen, si aislamos nuestra conducta y la observamos a partir de sus manifestaciones externas encontramos cinco funciones que se interconectan entre sí:

- Una función de **entrada**, constituída por nuestros sentidos, y por donde ingresan los datos desde el exterior a nuestro cerebro.
- Una función de **salida**, constituída por nuestra motricidad corporal y que es la respuesta a los datos del exterior.
- Una función de **almacenamiento** de los datos, constituída por nuestra memoria.
- Una función de **procesamiento** de los datos, que nos permite adecuar nuestra acción a la realidad, interpretando los datos.
- Una función de **control**, que ordena coordinamente las acciones de las funciones mencionadas.

Las palabras utilizadas para referirse a las funciones de las computadoras se han escogido, en su mayoría, de la vida común y corriente. Cosa que se puede prestar a confusiones.

Se dice que la computadora **memoriza**, que **resuelve** problemas, que podemos **conversar** con ella, etc.

Con el uso de estos términos, algunos se confunden al extremo de atribuir a las computadoras comportamientos humanos, o pseudohumanos.

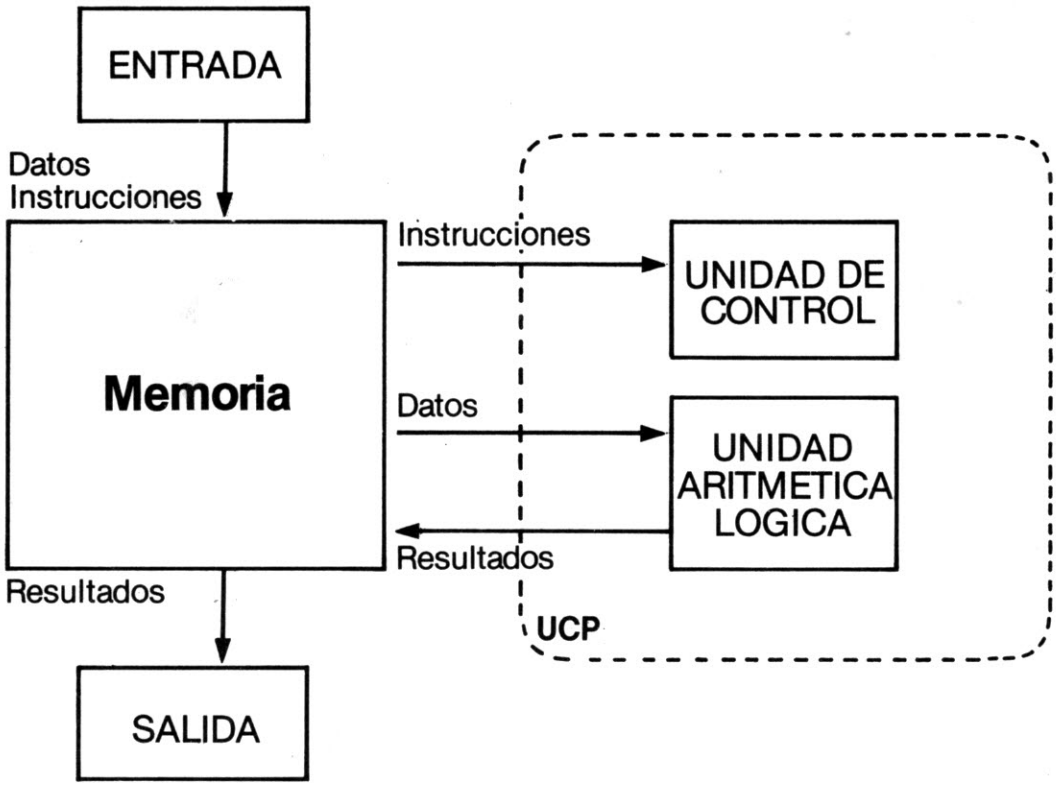
En cierto modo, la computadora memoriza, pero no hace **experiencia**. Nosotros recordamos nuestros errores para no repetirlos, mientras que los errores de las computadoras se repiten sin fin.

La computadora resuelve problemas, pero lo hace gracias a las instrucciones que le vamos entregando, de modo que si nos equivocamos, ella también se equivoca.

También veremos que existen "lenguajes" para "interrogar" a la máquina y que ésta nos "responde", y en otros casos nos "denuncia" un error y hasta nos "felicita" por un acierto.

A todo esto se lo llama "conversar" con la computadora. Un término sin duda exagerado, porque el **diálogo** entre personas (la conversación), es una de nuestras actividades más elevadas y quizás, la mayor fuente de enriquecimiento y desarrollo personal.

La Computadora, sus Partes y Funciones.



La computadora está construída según un modelo de funcionamiento muy sencillo y lógico. Este modelo es común a todas las computadoras u ordenadores, como también se las llama.

En el cuerpo de la computadora se distinguen dos partes:

- Una **externa**, constituída por sus dispositivos de entrada y salida. Por la primera entran datos e instrucciones, por la segunda salen resultados.
- Y otra **interna**, constituída por la memoria y la unidad central de procesamiento, denominada UCP. En la memoria se almacenan los datos e instrucciones y en la UCP es donde se producen los resultados y se controla el funcionamiento de la máquina, según sean las instrucciones.

Algo muy parecido, aunque muchísimo más simple, a nuestras funciones sensitivas, cerebrales y motrices donde:

- los sentidos y los miembros del cuerpo ejercen funciones de entrada y salida, respectivamente.
- y el cerebro es el lugar donde se almacenan las experiencias, se producen las conclusiones y se coordina el conjunto del organismo.

1. La Unidad Central de Proceso (UCP)

La UCP cumple dos funciones que se asemejan a las del cerebro, porque:

a. **Controla** y **coordina** a todas las demás unidades (como nuestro cerebro, que controla y coordina nuestros músculos voluntarios).

b. Realiza **operaciones aritméticas** (sumar, restar, multiplicar, etc.) y **operaciones lógicas** (comparar) con los datos.

La función **a.** está a cargo de la **Unidad de Control** y la función **b.** a cargo de la **Unidad Aritmético Lógica**.

A. Unidad de Control

Es la parte más compleja de toda la computadora. Estas son sus funciones específicas:

a. **Extrae de la memoria las instrucciones** a ejecutar (más adelante veremos cómo las instrucciones se almacenan en la memoria).

b. **Analiza la instrucción:**
Las instrucciones son de variados tipos. Por ej. pedir un dato al operador, entregar un dato a este mismo, extraer un dato de la memoria, etc.

c. Según sea la instrucción, **establece** determinadas **vinculaciones** en forma de conexiones electrónicas. Por ej.: si la instrucción es pedir un dato al operador, conecta la unidad de entrada, para ingresarlo; si se trata de ofrecer un dato, conecta la unidad de salida, etc.

d. **Extrae los datos** necesarios **de la memoria**, para ejecutar la instrucción.
Supongamos que en los pasos anteriores, la instrucción fue pedir al exterior (operador) el dato **a.** Ahora la instrucción es extraer de la memoria el dato **b.**, a fin de multiplicar ambos datos entre sí.

e. **Desencadena el tratamiento de los datos** en la unidad aritmético-lógica.
Siguiendo el ejemplo anterior, la unidad de control activa la unidad aritmético-lógica, para que se multiplique **a × b.**

f. **Almacena datos** y **resultados** en la memoria.
El dato **a.**, pedido al exterior, quedó almacenado y también el resultado **a × b**, quedó almacenado.

g. **Ordena la salida del resultado.**
El resultado **a × b** es exhibido al operador.

La unidad de control tiene que efectuar funciones diversas y complicadas. Primero, como ya dijimos, tiene que seleccionar una instrucción de la memoria. Luego analizar la instrucción para determinar la operación específica que se ha de realizar. A continuación ejecutar la operación. Durante la fase de ejecución, la unidad correspondiente opera bajo la dirección de la unidad de control.

El ciclo se repite: se extrae una instrucción, se analiza y se ejecuta. Cada ciclo completo se realiza a gran velocidad y tiene lugar como una serie de impulsos.

La unidad de control tiene que generar los impulsos, sincronizar y controlar las actividades de los numerosos circuitos y asegurarse que cada instrucción se realice en un período de tiempo determinado.

B. Unidad Aritmético-Lógica

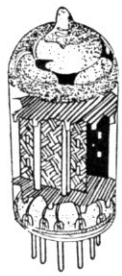
A su cargo está el **tratamiento de los datos**, operando sobre éstos según las órdenes que reciba de la unidad de control.

La unidad **aritmético lógica** está formada por una serie de circuitos especiales, que pueden realizar operaciones **aritméticas** y **lógicas** con uno o más operandos, seleccionados de la memoria.

Por ejemplo, pueden realizarse las operaciones de suma, resta, multiplicación y división, además de comparaciones que incluyen las operaciones lógicas "o", "y", "no", etc. (en inglés: "or", "and", "not", etc.).

Aspectos Electrónicos de la UCP:

El prodigio de los "circuitos integrados"



1ª GENERACION

2ª GENERACION

3ª GENERACION

La UCP es un circuito de integración a gran escala. Esta integración es un proceso de colocar millares de componentes en una superficie muy pequeña, constituyendo uno de los avances más asombrosos de la tecnología actual.

Dentro de la computadora, los impulsos eléctricos se controlan por piezas denominadas **componentes electrónicos**.

La primera generación de computadoras electrónicas usa válvulas o tubos al vacío, como componentes electrónicos. Recordemos la ENIAC, año 1946, que tenía 18.600 tubos al vacío y necesitaba un espacio de 150 m^2.

La segunda generación reemplaza los tubos por transitores. Las computadoras transitorizadas aparecen en 1959. Mucho más pequeñas y más confiables que las anteriores. Su tamaño ya permite la comercialización de las mismas.

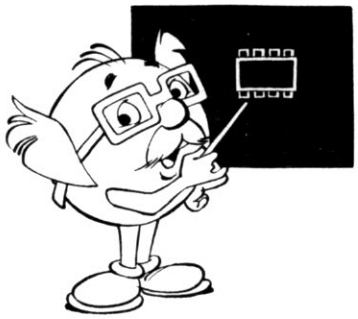
La diferencia de tamaño entre el tubo y el transistor, nos da una idea del achicamiento de las segundas RESPECTO DE LAS PRIMERAS COMPUTADORAS.

La tercera generación, que aparece en 1970, significa un salto prodigioso. Las decenas de miles de componentes electrónicos se incluyen en circuitos integrados que ocupan la superficie de un **chip**, que es el origen de los microprocesadores y de las microcomputadoras.

¿De dónde procede un chip y cómo se construye?

Procede de los **cristales de silicona**. Estos son compuestos análogos a los cuerpos orgánicos, donde el silicio reemplaza al carbono (Silicio, Si, metaloide de número atómico 14. Se funde a los 1.414°C).

El proceso de fabricación de un chip se puede desglosar en los siguientes pasos:



- Se cultivan cristales de silicona con una pureza de 99,9999999% en hornos al vacío. Por su pureza, la silicona no conduce la electricidad.
- La silicona se corta en capas, que parecen monedas muy delgadas.
- Se dibujan los circuitos que formarán parte del chip. Se dibujan a un tamaño 250 veces más grande que el chip.
- Se reduce el dibujo del circuito al tamaño del chip.
- El dibujo, así reducido, se fotografía sobre la superficie de la capa de silicona.
- La capa de silicona, con el diseño del circuito grabado fotográficamente, se coloca en un horno a más de 1.000 °C.
- Ya en el horno, la silicona es expuesta a ciertas sustancias químicas. Los átomos de esta sustancia penetran en la superficie de la silicona, pero sólo por las líneas impresas de los circuitos.
- Las líneas, impregnadas por la sustancia química, serán los conductores por donde fluirá la corriente eléctrica.
- Si el chip en cuestión ha de llevar más circuitos, se vuelve al proceso de: dibujo del otro circuito, reducción, fotografía, horno y exposición química. Algunos chips tienen once o más circuitos diferentes, conteniendo decenas de miles de componentes electrónicos, unos sobre otros.

2. La Memoria

Tiene dos funciones principales en la computadora:

a) **guardar datos** y b) **guardar instrucciones**.

Por lo tanto, podemos asemejarla a la función de almacenamiento del cerebro.

Los datos pueden ser cualquier cosa (precios, nombres, fechas...) depende del trabajo que deba ejecutar la computadora.

Las instrucciones son las órdenes que se le dan a la computadora, para que esta haga exactamente lo que hay que hacer.

Los datos son almacenados en forma de cargas eléctricas. La memoria contiene muchas células o posiciones que se encuentran en todo momento en los siguientes estados: positivo (+) o negativo (−).

Cada posición (célula) de la memoria está representada por un número único. La memoria de una computadora, por ejemplo, que tenga 1000 posiciones, podrá mantener 1000 caracteres. La memoria forma parte del circuito del chip, junto a la UCP.

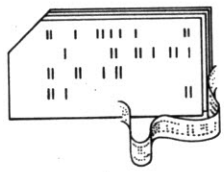
3. Unidad de Entrada

El dato que ingresa a la computadora procede del mundo real (nombres, dimensiones, precios, etc.) y debe ser colocado en forma aceptable para ésta.

La **unidad de entrada** es el medio por el cual las personas se comunican con la computadora, introduciendo los datos y programas.

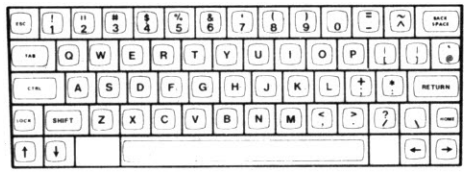
Es necesario disponer de alguna forma de codificación intermedia para llenar el vacío existente entre el lenguaje de los seres humanos y el lenguaje interno de las computadoras, cuyas memorias sólo pueden responder a combinaciones de impulsos eléctricos.

Estos son algunos de los **dispositivos de entrada** más comunes



- **Tarjetas y cintas perforadas**. Las perforaciones expresan el dato en forma codificada. Así, la información es "leída" por la computadora.

- **Teclado**. Es similar a una máquina de escribir. Un dispositivo especial decodifica la información, haciéndola aceptable para la computadora. Muy práctico y de uso corriente.

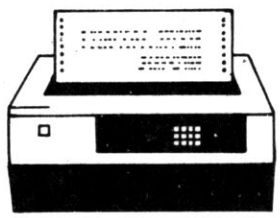


4. Unidad de Salida

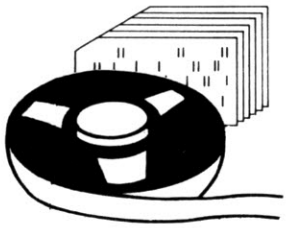
Con la salida culmina todo este proceso. De la máquina sale una información que necesitamos.

A partir de los datos que entraron y a partir, también de otros datos que están almacenados en la memoria, la computadora procesó una información que no teníamos, y que ahora debe salir.

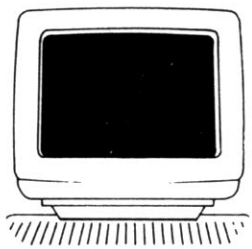
Existen distintos **dispositivos de salida**:



- **Impresora.** Nos entregan la información impresa en papel, con letras y números convencionales. Hay impresoras de gran velocidad.

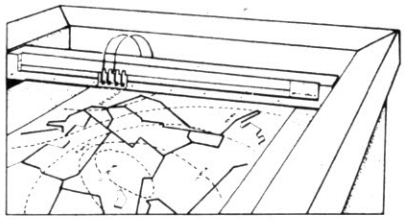


- Otros dispositivos transfieren la información a tarjetas, cintas de papel o cintas magnéticas, para otros tipos de usos.



- La pantalla tipo de televisor es un dispositivo de salida, muy práctico y de uso creciente.

- Trazadora de gráficos, dibujos y palabras tipo manuscritas. La computadora manda impulsos electrónicos a una pluma, girándola a través del papel. Algunos pueden cambiar de color automáticamente.



5. Almacenamientos Auxiliares

Son dispositivos que no forman parte de la computadora, pero se conectan con ella para cubrir dos necesidades:

1) aumentar la capacidad de almacenamiento de datos.
2) asegurar la permanencia de datos y programas, ya que éstos, de quedar en la memoria central, se perderían al apagar la computadora.

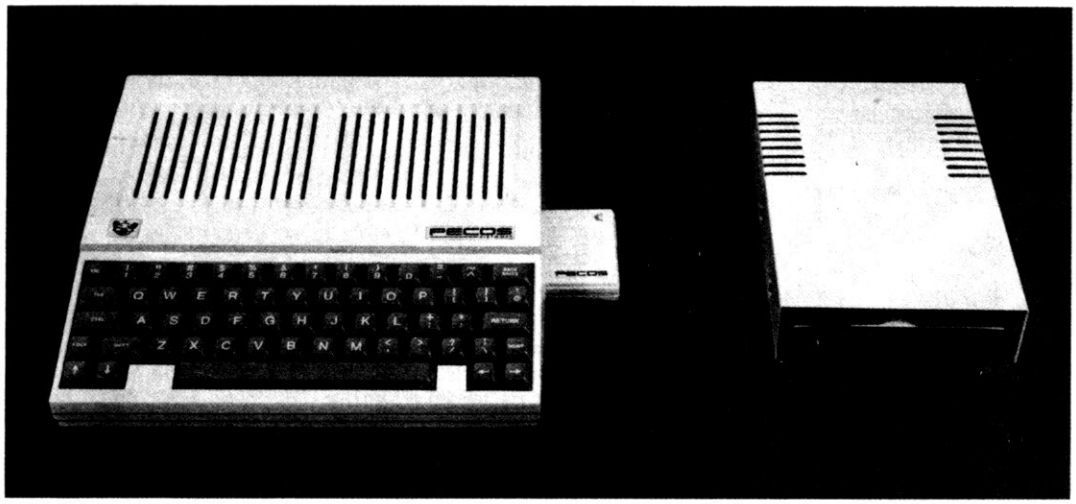
Los medios de almacenamiento más comunes son:

- **Cinta magnética**. Puede almacenar aproximadamente 10 millones de caracteres. La desventaja es que se necesita recorrer el carrete desde el comienzo, para recuperar el dato.

- **Disco magnético.** Utilizado cuando el dato debe ser recuperado rápidamente. Permite ser usado de a varios, formando una pila, donde el dato es recuperado mediante el acceso directo al disco que lo almacena.

- **Diskette**. Es un disco de plástico, flexible, con sensibilidad magnética en la superficie. La computadora puede obtener datos de cualquier parte del diskette.

- **Cassette**. También se puede almacenar en la cinta magnética de un cassette común.



Ahora conecten la Computadora, coloquen el DISKETTE, elijan la opción de CPM.
Luego escriban PECOS y toquen la tecla RETURN.
Para reforzar los conocimientos hasta acá obtenidos elijan la opción PECOS.

Aparecerán otras opciones con COMPONENTES, ver la estructura de la computadora y con INSTRUCCIONES repasar el flujo de cada instrucción que vieron.

Para ver lo que aprendieron elijan la opción PRUEBA.

Una vez que logren salir exitosos del "examen" pueden jugar un rato eligiendo la opción JUEGOS —y prueben de ganarle a Pecos—. Suerte!!!

El Sistema Binario

El sistema binario ha resultado ser el más adaptable y eficiente en el uso de las computadoras.

Recordemos que este sistema nos permite representar cualquier cifra, por grande que sea, utilizando solo dos números: el cero y el uno, de ahí su nombre de **binario**. También podemos simbolizar las letras del alfabeto con diversas combinaciones de ceros y unos.

Ahora bien, un sistema que ofrece una gran posibilidad de simbolizar, con solo dos elementos, nos permite aplicar todo un lenguaje a la máquina computadora.

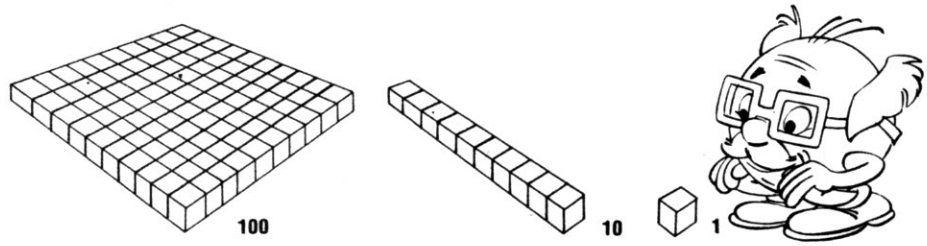
En efecto, todos los sistemas de almacenamiento que hemos visto, ofrecen dos alternativas. Una cinta de papel o tarjeta puede estar perforada o no perforada, pudiendo representar el 1 la sección perforada y el cero la no perforada. Un cassette puede emitir un sonido alto (representado por el 1) y un sonido bajo (representado por el 0). Un diskette, o un disco, puede estar magnetizado o no magnetizado en un punto de la pista. La corriente eléctrica puede pasar o no pasar por un circuito, etc.

Recordemos que Godfield Leibnitz, en el siglo XVII, demostró las ventajas del sistema binario sobre el decimal, para las computadoras mecánicas.

Repasaremos brevemente, los conceptos básicos del sistema binario.

Vamos a partir por lo conocido: nuestro sistema decimal. Para contar solo necesitamos 10 símbolos —0 y los dígitos del 1 al 9— pudiendo así escribir cualquier número, por grande que sea. Todos los números en este sistema son construídos en base a bloques valorados en 1, 10 y potencias de 10 (tales como 100, que equivale a 10^2; 1.000 que equivale a 10^3; etc.).

Como vemos en este cuadro, para construir el número 5, necesitamos cinco bloques de uno. Para construir el número 14, necesitamos un bloque de diez y 4 bloques de uno. 49 requiere cuatro bloques de diez y nueve de uno. Todo número decimal es un resumen de los bloques que se requieren para construirlo.

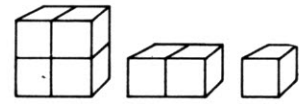


		5	5
	1	4	14
	4	9	49
			240

100.000	10.000	1.000	100	10	1	
10^5	10^4	10^3	10^2	10^1	10^0	
	1	0	4	3	9	10.439
						5.425
						126.428

Comprendido lo anterior ya no es necesario recurrir al dibujo de los bloques. El bloque de 1 cubo, es 10^0; el bloque de diez, es 10^1; el de 100, es 10^2, etc.

El sistema binario se expresa solamente con **dos símbolos**: 0,1. Utiliza bloques que valen potencias de 2 (tales como 1, que es 2^0; 2, que es 2^1; 4, que es 2^2; 8, que es 2^3, etc.). Al igual que el decimal, el binario es un sistema posicional —pero de base 2— donde los valores posicionales están escalonados según las potencias de 2.



64	32	16	8	4	2	1	
2^6	2^5	2^4	2^3	2^2	2^1	2^0	
					1	1	3
				1	0	1	5
			1	1	0	1	13
		1	0	0	1	1	
							126
1	0	0	1	0	1	0	

Construir el número 3 con bloques binarios requiere un bloque 1 y un bloque 2, por lo tanto se escribe 11. Escribir el 5 en forma binaria requiere un bloque 4, ningún bloque 2 y un bloque 1, escribiéndose 101.

El número 13 precisa un 8, un 4, ningún 2 y un 1, escribiéndose 1101. Completen el cuadro.

Los números binarios pueden ser fácilmente convertidos al sistema decimal, adoptando el método de notación posicional. A cada posición se le asigna el valor que le corresponde y luego se suman todos los valores obtenidos. Conviene comenzar por el dígito de mayor valor significativo.

Ej.:

Ej.: 1 11001

$1 \times 2^4 = 16$
$1 \times 2^3 = 8$
$0 \times 2^2 = 0$
$0 \times 2^1 = 0$
$1 \times 2^0 = 1$

25

$11001_2 = 25_{10}$

Ej.: 2 1101

1 × =
1 × =
0 × =
1 × =

$1101_2 =$

3) Traducir a su equivalente decimal: 10011

Sobre BITS y BYTES

Como ya dijimos, un sistema de computación está formado por elementos biestables (transistores, cintas, discos, cassettes, etc.) y los códigos formados de sólo dos señales se denominan códigos binarios. En Computación, se denomina BIT (abreviatura de la palabra inglesa BINARY DIGIT) a la representación por medio de un dígito binario de una situación real que puede tomar diferentes formas, por ejemplo, un punto magnetizable en una superficie de registro, un impulso en un circuito eléctrico, una pulsación o no pulsación en los circuitos de un chip, etc.

Casi todas las microcomputadoras utilizan conjuntos de 8 dígitos binarios para representar las unidades de información. Dicho octeto es denominado BYTE. Con un byte tenemos la posibilidad de representar 256 combinaciones distintas de ceros y unos.

Esto es suficiente para representar las letras del alfabeto, los números, los caracteres especiales que hay en un teclado, etc.

Como el byte resulta una unidad de medida, para expresar la cantidad de información que se puede almacenar en un disco, un diskette, etc, utilizamos múltiplos de él. Por ejemplo: 1024 byte se denominan "Kilobyte" o un K.

- Detectar un PROBLEMA.
- Plantear un ALGORITMO.
- Describir un PROGRAMA.
- Comunicarnos en un LENGUAJE compartido.

Estas son las etapas fundamentales del proceso del PENSAR humano.

PENSAR, CONOCER... para responder adecuadamente a las necesidades individuales y sociales. Para transmitir nuestra CULTURA y enriquecer espiritual y materialmente nuestra Sociedad.

La lectura de esta Unidad permitirá que los alumnos accedan al logro de los siguientes:

• Objetivos Generales

1. Se informen acerca de las características de este Problema: la Computación y el Software.
2. Que comprendan la naturaleza del proceso algorítimico.
3. Que comprendan la interacción que existe entre el aspecto estático y el dinámico de un Programa.
4. Adquieran información acerca de distintos tipos de Lenguajes.
5. Comprendan la relación entre Programa y Lenguaje.
6. Valoren el aporte de la Computación al proceso del Pensar, del Conocer humano.

Les proponemos ahora, como siempre, la lectura y el análisis del texto "El Programa", y su discusión en grupo, con la conducción de su coordinador, para acceder al logro de los Objetivos Generales propuestos y al de los siguientes :

• Objetivos Específicos

1.1. Enumeren los elementos del Software.
1.2. Definan el concepto "Instrucción".
1.3. Describan un Programa.
2.1. Analicen el proceso Algorítmico.
2.2. Expliquen las limitaciones y posibilidades de una Computadora.
3.1. Expliquen la relación que existe entre la tarea del Programador y la UCP.
4.1. Enuncien la característica fundamental del Lenguaje de Máquina.
4.2. Caractericen los Lenguajes Ensambladores.
4.3. Caractericen los Lenguajes de alto nivel teniendo en cuenta las ventajas que proporcionan para el trabajo del Programador.

4.4. Citen distintos Lenguajes de alto nivel.

4.5. Informen acerca del significado de las siglas con que se nombran los Lenguajes mencionados en 4.4.

5.1. Expliquen la relación que existe entre Programa Fuente, Programa Traductor, y Programa Objeto.

5.2. Informen acerca de la semejanza y diferencias entre los tres tipos de Programas Traductores: Ensambladores, Compiladores, Intérpretes.

6.1. Participen en los grupos de aprendizaje formulando preguntas significativas y pertinentes a los aspectos estáticos y dinámicos de un Programa.

6.2. Juzguen la importancia que asume la buena definición de un Problema para su resolución.

6.3. Juzguen la importancia que asume el correcto planteo de un Algoritmo para la posible solución de un Problema.

6.4. Valoren los Lenguajes creados por el hombre como instrumentos para su actividad más completa: la Comunicación.

6.5. Intenten autoevaluar este Aprendizaje en el propio proceso del Conocer, del Pensar.

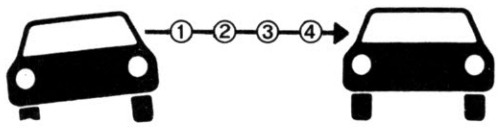
Introducción

En la unidad anterior (N° 2) —La Máquina— analizamos las principales partes de la computadora, sus funciones específicas y los materiales de que están hechas.

Se ha extendido, entre los profesionales de la computación, el término **hardware** para denominar lo que, hasta ahora, hemos llamado simplemente "máquina". Aquel término proviene del inglés **hard**, que significa duro, es decir, la parte "dura" de la computadora. Literalmente, hardware significa "ferretería", de ahí que la jerga profesional utiliza también el término fierros, para referirse a la máquina.

Hardware es un concepto opuesto a **software**. Este término se refiere a los contenidos de alimentación de la máquina, con todas sus implicancias (programas y datos, lenguajes, intérpretes, etc.) Es lo que veremos a continuación.

1. El Programador, el Procesador y el Programa



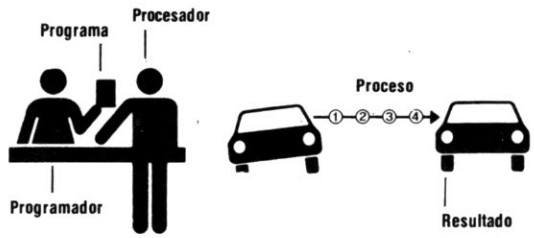
Para resolver un problema necesitamos encontrar una sucesión finita de pasos que nos provea la solución del mismo.

En este ejemplo, para pasar de la situación "goma pinchada" a la de "goma normal", el automovilista debió realizar una serie de acciones a la que llamamos **proceso** ("acción de ir hacia adelante", del latín proccesus). Este proceso lo desarrolla a través de pasos, cuyo orden y sucesión debe respetar (por ej.; afloja las tuercas, aplica el criquet, levanta la rueda; y no: aplica el criquet, levanta la rueda y afloja las tuercas; etc.).

Si le pedimos al automovilista que ponga por escrito, los pasos sucesivos que cumplió para cambiar la rueda, tenemos un **programa**. Y cada paso será una **instrucción**, para el inexperto que desee cambiar una rueda correctamente.

Entonces tenemos que un programa es una secuencia de instrucciones, escritas por un programador.

Quien ejecute las instrucciones —en el ejemplo, el automovilista inexperto— será el **procesador**. Este es quien recorre los distintos pasos del proceso de cambiar la rueda.



El programa es en sí algo estático y el proceso de ejecutar las instrucciones es algo dinámico.

La computadora es una máquina, carece de imaginación y sentido común. Pero, si hacemos un programa donde las instrucciones se presentan de un modo claro y sin ambigüedades, puede resolver problemas por sí misma. Dicho en pocas palabras, puede ser un procesador, que ejecute las instrucciones y por lo tanto lleve a cabo un proceso.

En la unidad N° 2 —la máquina— vimos que la UCP (unidad central de proceso) cumple, precisamente, con la función del procesador. La Unidad de Control extrae una instrucción del programa (que está almacenado en la Memoria), la analiza y la ejecuta, activando las unidades de la máquina que correspondan.

— ¡Qué manera de pedir una morcilla!

Un programa, es el medio por el cual el programador **se comunica** con el procesador.

Pero toda comunicación requiere de un **lenguaje**. Por lo tanto, el programador deberá poner sus instrucciones, en un lenguaje que el procesador **comprenda**.

En el caso de la goma pinchada, si las instrucciones están escritas en un idioma que el automovilista inexperto ignora (por ej. coreano) ese programa no le sirve.

Cuando el procesador es una computadora, el problema del lenguaje es absolutamente más severo, puesto que no tiene los recursos del ser humano para salir del paso (imaginación, sentido común, etc.).

2. Sistema Binario y Lenguaje

El sistema binario es el lenguaje de la computadora.

Esto sigue siendo así, por más que se progrese en los puentes que el hombre sigue ideando para cubrir las distancias entre el lenguaje humano y el sistema binario.

En los primeros tiempos, había que almacenar el programa en la memoria de modo tal, que cada instrucción era una serie de ceros y unos.

Veamos un ejemplo de como podía ser representada la instrucción de sumar dos números, en este lenguaje de máquina.

Toda operación aritmética tiene su correspondiente código. Por ejemplo, la instrucción de sumar se expresa en el código 0101.

Ahora, si se trata de sumar dos operandos, cada uno de ellos tiene, si se nos permite la expresión, su propio "domicilio" dentro de la memoria. A este "domicilio" se le llama dirección y cada dirección tiene su propio código. Entonces a la Unidad de Control le ordenamos que vaya a la dirección tal de la memoria y extraiga el número que allí está almacenado (algo así como el cobrador de cuotas de un club, que va a la dirección de una casa y "extrae" el dinero correspondiente al mes que se debe). Luego le daremos otra dirección a la unidad de control para que extraiga el segundo operando de la memoria y luego lo sume con el anterior. Al resultado le asignaremos su propio "domicilio", su propia dirección de memoria con el correspondiente código.

Entonces, la instrucción podría ser:

a) Código de la operación (suma, resta, etc.).
b) La dirección de la posición de memoria en la que se encuentra el primer operando.
c) La dirección de la posición de memoria en la que se encuentra el segundo operando.
d) La dirección de la posición de memoria en que se almacena el resultado.

Ahora le asignamos, a cada uno de estos cuatro pasos, su correspondiente código.

0101	0111	1001	1101
a)	b)	c)	d)

La instrucción de sumar, como ya vimos, se expresa en el código 0101.

Dicha instrucción, entonces, ordena sumar el contenido de la dirección 0111, con el contenido de la dirección 1001 y el resultado alamcenarlo en la dirección 1101.

Esta es una instrucción de tres direcciones.

Es fácil imaginar que la programación en **lenguaje de máquina** no es cómoda e implica una tarea muy pesada para el programador. Particularmente requiere grandes esfuerzos de memorización.

Un nuevo paso lo constituye el reemplazo del código binario de operación por grupos de letras mnemotécnicas (para facilitar la memorización de la operación por parte del programador), dando lugar a un lenguaje llamado **ensamblador**. Así el código de operación de la suma se representaba por ADD, el de la resta por SUB, etc.

Pero las instrucciones así escritas guardan todavía demasiada relación con el lenguaje de máquina.

Surgen entonces los llamados **lenguajes de alto nivel**, que a diferencia de los ensambladores (lenguaje de bajo nivel) son más parecidos al lenguaje humano y mucho más prácticos para la tarea de programar.

Los lenguajes de alto nivel son universales, la persona solo tiene en cuenta el problema a resolver y se despreocupa de la estructura interna de la computadora.

Entre los lenguajes de alto nivel tenemos:.

- FORTRAN (Formula Translation).
- COBOL (Common Bussiness Oriented Language).
- ALGOL (Algorithmic Language).
- PL/I.
- PASCAL.
- BASIC (Beginner's All purpose Symbolic Instruction Code).
- ETC.

Ahora se plantea el problema de traducir el lenguaje de alto nivel al lenguaje de máquina, el único que la computadora puede ejecutar. Es necesario, entonces, dotarla con programas **traductores**.

Se llama **programa fuente** al escrito en lenguaje evolucionado y **programa objeto** al que resulta de traducirlo al lenguaje de máquina.

Hay tres tipos de programas traductores:

- **Ensambladores**, que se usan para traducir programas en ensamblador. Toda instrucción del programa fuente, da lugar a una sola instrucción del programa objeto.

- **Compiladores**, que se usan para traducir programas escritos en lenguajes de alto nivel. Una instrucción del programa fuente se traduce a varias del programa objeto.

- **Intérpretes**, tiene los mismos fines y características que los compiladores. Pero, mientras éstos traducen todo el programa que luego se va a ejecutar, los intérpretes traducen una instrucción, la ejecutan y luego continúan con la siguiente.

3. El Algoritmo

El término "algoritmo" es muy antiguo entre los matemáticos. Proviene del sabio árabe al-Khwārizmi que vivió en el Siglo IX y contribuyó, desde España, a la civilización de Europa (cfr. Unidad Historia, N° 1, pág. 2).

El término **algoritmo** se refiere al método y notación en las distintas formas de cálculo. Este es su sentido más antiguo y originario. Los científicos de la computación tomaron este término y lo aplicaron a su propia actividad.

En computación, antes de escribir un programa (compuesto de instrucciones precisas, siguiendo un orden preciso y con un lenguaje determinado, según la máquina), el programador debe plantearse un **método** que le permita ir definiendo la solución del problema, **paso por paso**. En otras palabras, un programador no puede redactar un programa para la máquina, si antes no tiene su propio **algoritmo**.

Siguiendo el ejemplo de la goma pinchada, nadie puede redactarle un programa de recambio de goma a un automovilista inexperto, y menos en su propio idioma, si el mismo redactor no tiene bien definido el problema ni sabe como abordarlo, paso por paso. Quien sabe cambiar una goma pinchada tiene ya su propio algoritmo. Bastaría que escribiera los distintos pasos que se necesitan para desarrollar la tarea y tendría su algoritmo por escrito. Luego, sobre esta base, tendría que explicitar, una por una las instrucciones para el automovilista inexperto y tendría su programa. Finalmente y dado el caso, habría que traducirlo al idioma del aprendiz.

Un algoritmo es la especificación de una sucesión finita de pasos que, si se ejecutan en el orden prescripto un número finito de veces, se obtiene la solución del problema o se encuentra que no tiene solución. Y en ambos casos, la ejecución se detiene.

•

Ejemplo:

Calcular el valor del número entero positivo **x** que satisface la ecuación **a** + **x** = **b** para dos números enteros positivos **a** y **b**.

El algoritmo podría ser:

1. Comparar el valor de **a** con el de **b**. Si **b** es mayor que **a**, continuar con el paso siguiente. Si **b** es igual que **a** o es menor que **a** parar la ejecución, e informar que no hay solución (la ecuación, no tiene solución entera positiva).
2. Definir **i** igual a **1**.
3. Sumar **a** + **i** y obtener **s**.
4. Comparar **s** con **b**.
   Si son iguales, detener la ejecución; la solución **x** tiene el valor de **i**.
   De lo contrario, sumar **1** a **i** y volver al paso 3.

Es importante destacar que el algoritmo es válido **para cualquier valor** de **a** y **b**, siempre y cuando estos sean enteros positivos.

El algoritmo, además de ser un conjunto finito de pasos para resolver un problema específico, tiene cuatro características importantes: finitud, exactitud, entrada y salida.

Veamos que significa cada una.

- **Finitud**. Un algoritmo debe terminar siempre después de un número finito de pasos.
  En el algoritmo dado como ejemplo se satisface esta condición, ya que, si existe una solución entera positiva de la ecuación **a + x = b**, el algoritmo la encontrará después de un número finito de pasos, cuando **s** sea igual a **b**.
  Y si no hay solución entera positiva, esto se descubrirá en el paso **1**.
  Notemos, sin embargo, que el número de pasos puede volverse arbitrariamente grande si elegimos valores de **a** pequeños y valores de **b** enormes.
- **Exactitud**. Cada paso de un algoritmo debe ser definido en forma precisa. Para cada caso, la acción a realizarse debe ser especificada de manera rigurosa y sin ambigüedad.
  El paso **1** del ejemplo, supone que la persona que lo lee entiende exactamente que significa comparar un par de números enteros positivos y reconocer cuando un entero positivo es mayor que otro.

- **Entrada**. El algoritmo puede tener cero o más entradas, esto es, cantidades que son dadas inicialmente, antes que el algoritmo comience.
  En el ejemplo hay dos entradas: los números **a** y **b**.
- **Salida**. Un algoritmo tiene una o más salidas, esto es, cantidades que tienen una relación específica con las entradas.
  El ejemplo tiene una salida **i**, que es el valor del número entero positivo **x**, que satisface la ecuación **a + b = x**.

En síntesis, desde la producción de un programa hasta la ejecución del mismo, se requiere pasar por las siguientes etapas:

- Planteamos el algoritmo en términos de nuestro propio idioma, en este caso, el castellano.
- Con el algoritmo —escrito en castellano— escribimos el programa fuente, en un lenguaje determinado. Esto dependerá del tipo de máquina a la que van dirigidas las instrucciones.
- El programa lo ingresamos a la computadora y el programa traductor, lo traduce al lenguaje de máquina.

El lenguaje del programa fuente debe escribirse observando y respetando reglas muy precisas y rigurosas (incluyendo la ortografía, la concordancia entre el género del sustantivo y el género del adjetivo, etc.) Cuando estas reglas se transgreden, el programa traductor acusa el error y se detiene el proceso.

Ejercicios

1. Determinen cuál de los siguientes procedimientos es un algoritmo. Justifiquen cada respuesta. En los casos afirmativos digan que problema resuelve.

a) PASO 1
- Comienzo
- Busque su licuadora
- Colóquela sobre la mesa
- Continúe paso 2

PASO 2
- Tome un litro de leche
- Viértalo en la licuadora
- Continúe paso 3

PASO 3
- Tome 100 grs. de azúcar
- Viértalo en la licuadora
- Continúe paso 4

PASO 4
- Tome 4 bananas
- Colóquelas sobre la mesa
- Continúe paso 5

PASO 5
- Si tiene bananas en la mesa continúe paso 6 sino continúe paso 7

PASO 6
- Tome una banana
- Pártala en trozos
- Coloque los trozos dentro de la licuadora.
- Vuelva al paso 5

PASO 7
- Tape la licuadora
- Enchúfela
- Póngala en marcha
- Déjela funcionar durante 5 minutos
- Continúe paso 8

PASO 8
- Vierta el licuado en un vaso
- Saboréelo
- Fin

b) A1- • Comienzo
- Tome una hoja en blanco
- Colóquela en su banco
- Continúe A2

A2- • Tome un lápiz
- Continúe con A3

A3- • Si la hoja está escrita vaya a A7
- Sino continúe con A4

A4- • Escriba su nombre
- Continúe con A5

A5- • Tome una goma
- Borre lo escrito
- Deje la goma en el banco
- Continúe A6

A6- • Vuelva al paso A3

A7- • Fin del proceso

c) B1- • Comienzo
- Tome una cierta cantidad de carne
- Colóquela sobre la parrilla
- Continúe B2

B2- • Tome la sal
- Coloque a gusto sobre la carne
- Deposite la sal
- Continúe con B3

B3- Fin del proceso

2. Dados los siguientes problemas, escriban los algoritmos que resuelven los mismos.
   a) Calculen la hipotenusa de un triángulo rectángulo conociendo sus catetos.
   b) Dado un número real calculen su valor absoluto.
3. En los problemas del ejercicio anterior, expliciten cuál es la entrada y la salida de cada algoritmo.

Al elaborar las tres unidades anteriores abordamos el estudio de la Computación y el de las Computadoras tratando de ubicarlo en el contexto general del pensar y el conocer.

Ahora nos proponemos presentar a Uds. un lenguaje apto para programar los algoritmos correspondientes a un problema específico.

Recordemos que un lenguaje es el instrumento indispensable para la comunicación, actividad que define al ser vivo como tal. Existen distintos tipos de lenguaje y necesitamos de ellos para vivir y convivir: el corporal, el mímico, y el verbal hablado y escrito. En todos ellos reconocemos elementos comunes porque todo lenguaje es un sistema, una estructura cuyo código y significado compartimos.

Existen distintas funciones del lenguaje: expresiva, informativa, directiva.

El lenguaje de la Computación se caracteriza por sus funciones informativa y directiva (Afirma o niega proposiciones. Origina o impide una acción manifiesta). Excluye la función expresiva no sólo porque se trata de un lenguaje científico sino especialmente porque el intermediario de la comunicación entre los hombres es, esta vez, una máquina, que no sólo ayuda a operar sino también a controlar.

El lenguaje que hemos seleccionado para que Uds. ingresen en el area de la comunicación, con y a través de la computadora, se llama TIMBA. Fue creado por un equipo de trabajo de la Universidad Nacional de San Luis, dirigido por el Ing. Hugo Ryckeboer, "...como respuesta a la necesidad de contar con un lenguaje sencillo de programación...", que permita estructurar programas trabajando con pilas de cartas. Hemos seleccionado este lenguaje de programación pues si bien es uno de los más sencillos ofrece gran posibilidad didáctica. Esto es, permitirá a Uds. acceder al conocimiento, comprensión y aplicación de otros lenguajes de Computación más ricos y complejos.

Recomendamos especialmente la revisión de la Unidad III antes de empezar esta tarea.

Para evaluar el aprendizaje logrado por Uds., luego de estudiar esta Unidad en grupo con la conducción de su coordinador, hemos formulado los siguientes:

• Objetivos Generales

1. Conozcan Uds. las características del lenguaje TIMBA.
2. Conozcan la estructura de un programa escrito en lenguaje TIMBA.
3. Conozcan la simbología utilizada en la descripción de las sentencias.
4. Comprendan la simbología utilizada.
5. Conozcan las funciones de las sentencias.
6. Comprendan el significado de "condición".

7. Conozcan las palabras reservadas del lenguaje TIMBA.
8. Desarrollen programas adecuados para resolver problemas.
9. Comprendan la interrelación que existe entre el Pensar y el Hacer.
10. Valoren la importancia de los conocimientos adquiridos al estudiar las unidades anteriores.

• Objetivos Específicos

1.1. Describan las operaciones posibles con una pila de naipes en lenguaje TIMBA.

2.1. Mencionen las partes que componen un programa.

2.2. Describan la forma de expresar la definición de un proceso algorítmico.

2.3. Clasifiquen los distintos tipos de sentencias, órdenes o instrucciones que se formulan en un proceso algorítmico.

2.4. Ubiquen la "descripción de los datos de la pila" en nuestro proceso de comunicación con la Máquina.

3.1. Nombren los elementos que constituyen la simbología utilizada en la descripción de sentencias.

3.2. Describan distintas formas de escribir una sentencia.

4.1. Analicen la descripción de los datos de las pilas, en función de la simbología estudiada.

5.1. Definan el concepto de "sentencia".

5.2. Clasifiquen los distintos tipos de sentencias por su función.

5.3. Describan cada una de las funciones mencionadas.

5.4. Expresen formalmente cada uno de los tipos de sentencias mencionadas.

6.1. Escriban una proposición lógica (simple o compuesta), que emita un juicio sobre el estado de las pilas o de la carta y que pueda ser evaluado como verdadero o falso.

7.1. Mencionen las palabras reservadas en TIMBA que no pueden ser usadas como nombres de las pilas.

8.1. Resuelvan problemas desarrollando e implementando programas TIMBA adecuados.

8.2. Sepan interpretar rápidamente los mensajes emitidos por UCP al detectar un error.

9.1. Relacionen las conductas necesarias para programar con aquellas generadas al ejecutar y recibir respuesta a través de la Computadora.

10.1. Participen activamente en la interpretación de aquellos mensajes emitidos por la Máquina, que salen del contexto del lenguaje TIMBA, en función de conocimientos adquiridos en las Unidades anteriores.

10.2 Contribuyan a esclarecer las funciones de la Computadora, trascendiendo el preconcepto de su naturaleza "mágica".

Introducción

Timba es un lenguaje de programación que maneja pilas de cartas.

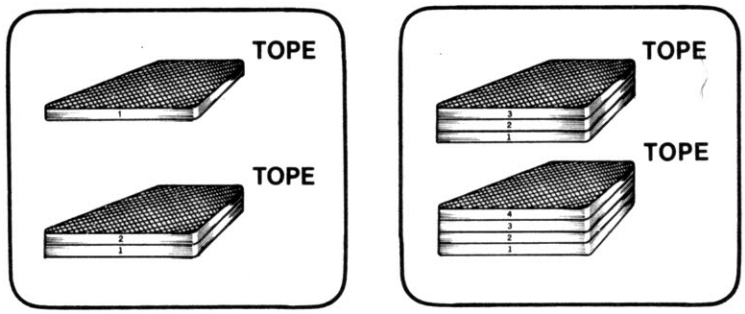
Todos tenemos una noción de lo que es una **pila**: pila de platos, de ladrillos, de cartas, de libros...

En todos los casos, unos están encima de los otros, como formando una torre, a la que llamaremos: pila.

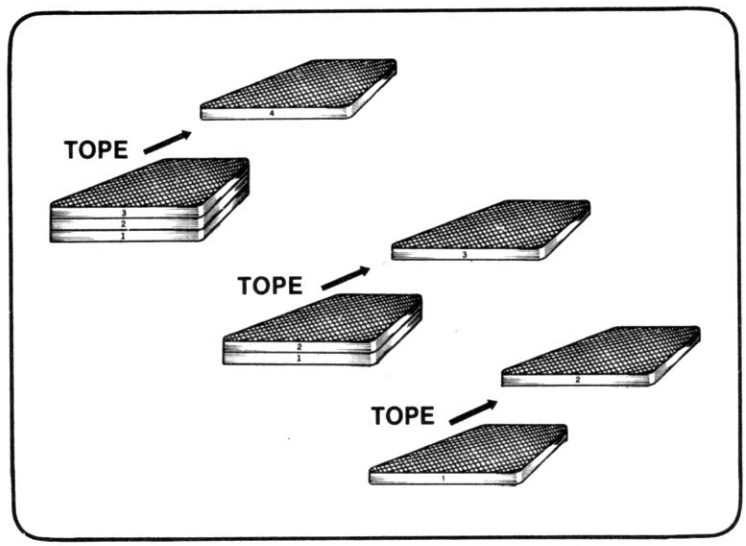
Ahora nos ocuparemos de **pilas de naipes**.

Una pila, salvo que esté vacía, tendrá siempre una carta por encima de las demás. Esta carta, aunque sea una sola, se llamará **tope de pila**.

Cada carta que se incorpore a la pila se debe colocar sobre el tope, y al hacerlo se convierte en el nuevo tope de la pila.



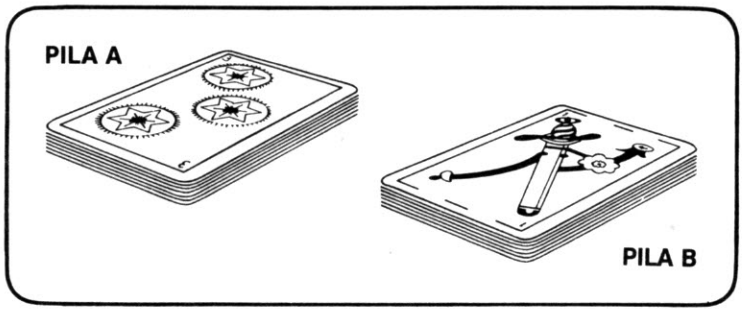
Al retirar cartas de una pila estamos restringidos a hacerlo de a una por vez. La carta que retiramos es, siempre, el tope. Y entonces la carta que estaba debajo pasa a ser el nuevo tope de la pila.



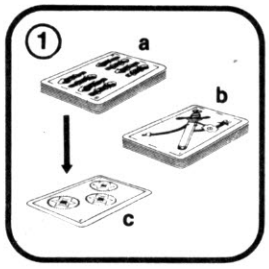
Un programa en TIMBA es una secuencia de instrucciones (u órdenes) escritas por un programador y dirigidas a un ejecutor.

Tomemos un mazo de cartas españolas y formemos dos pilas al azar.

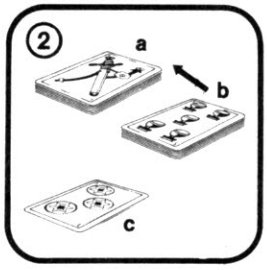
—La pila A tiene como tope de pila el 3 de oros.
—La pila B tiene como tope de pila el 1 de espadas.



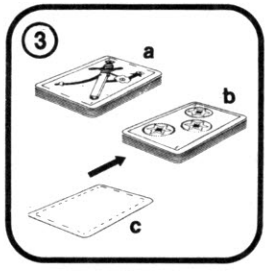
Problema: dadas dos pilas de cartas, la A y la B, intercambiar los topes de pila; de suerte que el tope de la pila A (3 de oros) pase a ser tope de la B y el tope de la B (1 de espadas) sea de la A.
Algoritmo: (resolución del problema).



1. Tome de pila A y deposite formando pila C



2. Tome de pila B y deposite en pila A



3. Tome de pila C y deposite en pila B

Los problemas podrían complicarse: dadas tres pilas: A, B y C, intercambiar los topes de pilas.

1. Nociones Generales de Timba

Hemos planteado un sencillo problema en términos de algoritmo, en castellano, usando palabras y conceptos comunes. Para pasar a un "programa", tenemos que adoptar un lenguaje de programación, que tiene un alto grado de rigor y precisión, como veremos.

Al **ejecutor** del programa lo llamaremos UCP. Es el responsable de los **resultados** y del **análisis de los errores**.

Una máquina computadora capaz de "comprender" el lenguaje TIMBA, puede ejecutar un programa, ya sea escribiendo en la pantalla las instrucciones o dibujando los naipes (en una dibujadora de gráficos o en la misma pantalla). En los juegos de ejercitación puede también una persona tomar el rol de UCP, ejecutando el programa mediante la directa manipulación de los naipes.

¿Cuáles son las órdenes que UCP reconoce en el lenguaje TIMBA?

En el proceso de ejecutar las instrucciones, UCP reconoce:

a) Las pilas por su nombre (Por ej.: PILA A, PILA B, etc.).
b) Las órdenes correspondientes a las operaciones de: "tomar", "depositar" e "invertir" cartas.
c) Las estructuras de control (esto lo veremos más adelante. Son ciertos mandátos condicionales, que permiten controlar las operaciones señaladas en el punto b).
d) Por último, UCP reconoce un ente especial llamado "CARTA" o "LA CARTA" y que se refiere, implícitamente, a cualquiera de los naipes con los que debe operar en un determinado momento.

2. Estructura de un programa en TIMBA

Un programa TIMBA está formado por dos partes:

— la definición del proceso algorítmico y
— la descripción de los datos de las pilas.

a) Definición del proceso algorítmico.

La definición del proceso algorítmico consiste en la formulación de las instrucciones, órdenes o sentencias (de ahora en adelante las llamaremos **sentencias**).

En la definición del proceso escribimos todas las sentencias, separadas entre sí por una coma (,). Después de la última sentencia colocamos un punto (.), que nos indica la finalización del proceso algorítmico.

UCP "comprende" tres tipos de sentencias:

1. operativas
2. de selección
3. iterativas

1. Sentencias operativas:

Las sentencias operativas son las que **actuan** sobre las pilas o la carta.

Comienzan con un verbo imperativo, que es interpretado por UCP como una orden.

Los verbos posibles son:

TOME
DEPOSITE
INVIERTA

Por ejemplo:

TOME UNA CARTA DE LA PILA A,
INVIERTA LA CARTA,
DEPOSITE LA CARTA EN LA PILA B.

2. Sentencias de selección

Nos permiten solicitarle a UCP que evalue una situación y, en función de ella, que adopte **alternativas**. Todo ello dependerá de ciertas condiciones, que UCP verificará durante la ejecución.

Una sentencia de selección comienza con el adverbio "SI", después le sigue una condición y, finalmente, las sentencias que deben ejecutarse en caso de ser verdadera la condición.

Si no hubiera dos secuencias alternativas, sino sólo un bloque de sentencias que deben ejecutarse en caso de cumplirse la condición o saltearlas si no se cumpliera, entonces la sentencia SI finaliza con las palabras SINO NADA MAS. Si hubiera una secuencia a realizar en caso de no cumplirse la condición y sólo entonces, la secuencia la debemos incluir entre las palabras SINO y NADA MAS.

3. Sentencias iterativas

(Iterarivo,va. Adj.: que tiene la condición de reiterarse. Diccionario de la Real Academia Española).

Estas sentencias nos permiten ordenarle a UCP que realice en forma **repetida** un bloque de sentencias, el cual estará gobernado por una condición que UCP verificará durante la ejecución.

Se las llama también sentencias MIENTRAS, porque siempre están precedidas por dicho adverbio.

Una sentencia iterativa comienza siempre con el adverbio MIENTRAS, después le sigue una condición, las sentencias que deben ejecutarse en caso de ser verdadera la condición y finalmente la palabra REPITA.

Resumiendo:

La primera parte de un programa en TIMBA está formada por el **proceso algorítmico**.

El proceso algorítmico está formado por sentencias (instrucciones) que son órdenes de ejecución para UCP.

Las sentencias son de tres tipos:

— operativas (TOME, DEPOSITE, INVIERTA)
— de selección (SI... SINO... NADA MAS)
— iterativas (MIENTRAS... REPITA)

Las sentencias operativas realizan cambios sobre la configuración de las pilas o de la carta que UCP tiene en ese momento, lo que se simboliza "por la carta que UCP tiene en la mano".

Las sentencias de selección y de iteración, en cambio, no alteran la configuración de las pilas ni de la carta.

Ambas son **sentencias de control** que implican órdenes condicionales. Permiten controlar las secuencias de operaciones, sin ser operaciones en sí.

b) Descripción de los datos de las pilas.

Una vez que escribimos el proceso, si lo queremos ejecutar, le debemos dar a UCP una descripción de las cartas que están en cada una de las pilas nombradas en el proceso.

Las pilas pueden estar vacías o no vacías. Si queremos que la pila este vacía, cuando UCP nos pide que le describamos el tope de la pila escribimos "NINGUNA", en caso contrario describimos la carta que queremos como tope y luego las cartas que queremos debajo, cuando no queremos más cartas en la pila ponemos "NINGUNA".

Por ejemplo:

1 DE OROS
4 DE BASTOS
7 DE OROS
12 DE COPAS
NINGUNA

Más adelante veremos que en la descripción de las pilas se especifica si las cartas están boca arriba o boca abajo.

3. Simbología utilizada en la descripción de sentencias.

En la descripción formal de las sentencias TIMBA, aparecen:
— corchetes [ ]
— llaves { }
— asteriscos *
— palabras claves de TIMBA
— los signos de puntuación de TIMBA
— algunas palabras en minúscula

Las palabras que aparecen en mayúscula en una descripción formal son palabras claves de TIMBA, y por lo tanto las debemos escribir tal como aparecen.

Las palabras en minúscula debemos reemplazarlas por su definición.

Por ejemplo, veamos la descripción formal de una sentencia de selección SI:

SI condición sentencias { SINO sentencias NADA MAS | SINO NADA MAS }

Esta descripción, según vemos, contiene las palabras claves SI, SINO, NADA, MAS.

Cuando queremos escribir una sentencia de selección SI, debemos empezar por la palabra clave SI, escribir a continuación lo que se describe en CONDICION como una condición y luego las sentencias que deseemos.

Puesto que "condición" y "sentencias" están escritas en minúsculas en la descripción formal, las reemplazamos por otros elementos más sencillos. La excepción es la palabra "nombre", que cuando aparece en una descripción formal debemos reemplazarla por una cadena de hasta diez caracteres alfabéticos o numéricos que no sea una palabra clave de TIMBA, por ejemplo MIL, 1000, XL.

Siguiendo con el ejemplo de la sentencia de selección, veamos el uso de las llaves: cuando en una descripción formal aparecen dos o más ítems entre llaves, debemos elegir entre ellos aquel que se ajuste a nuestro problema.

En el ejemplo de la sentencia SI, debemos elegir entre dos alternativas:

1 - escribir la sentencia SI con la opción: SINO sentencia NADA MAS

2 - o escribirla con la opción: SINO NADA MAS.

En el primer caso deberemos elegir las sentencias que incluiremos, y en el segundo deberemos copiar las palabras claves en el mismo orden en que aparecen.

Cuando en una descripción formal aparece un ítem entre corchetes podemos optar por escribir o no ese ítem sin que por ello UCP modifique la interpretación que hace del programa.

Por ejemplo, la descripción formal de una sentencia TOME.

TOME [UNA][CARTA] DE [LA] PILA nombre

Tiene tres pares de corchetes, cada uno de los cuales encierra **UNA**, **CARTA** y **LA**.

Esto significa que podemos escribir una sentencia TOME en 8 formas:

TOME DE PILA xxx
TOME DE LA PILA xxx
TOME CARTA DE LA PILA xxx
TOME CARTA DE PILA xxx
TOME UNA DE LA PILA xxx
TOME UNA CARTA DE PILA xxx
TOME UNA DE PILA xxx
TOME UNA CARTA DE LA PILA xxx

y todas estas formas son indistintas para UCP.

Por último, si un ítem se encuentra entre corchetes seguidos por un asterisco, podemos omitir el ítem o escribirlo una, dos o más veces, de acuerdo a nuestras necesidades.

Por ejemplo:

$$\text{Proposición}\left[\left\{\begin{array}{c}\text{Y}\\ \\ \text{O}\end{array}\right\}\text{proposición}\right]^{*}$$

Significa que podemos escribir una, dos o más proposiciones vinculadas por los operadores lógicos "Y" u "O".

4. Sentencias

Las sentencias son el elemento constitutivo de un programa TIMBA, una sentencia en TIMBA es la mínima unidad lógica.

Las sentencias en TIMBA pueden ser: operativas o de control.

Las **sentencias operativas** realizan cambios sobre la configuración de las pilas o de la carta que analiza UCP en ese momento, que sería "la carta que UCP tiene en la mano".

Las **sentencias de control** son comandos condicionales, y en sí no alteran la configuración de las pilas ni la de la carta, sino que son las sentencias que a su vez contienen las que lo hacen. Las sentencias de control nos permiten controlar la secuencia de operaciones, sin ser operaciones en sí. Hay tipos de sentencias de control: de **selección** (SI) y de **iteración** (MIENTRAS).

Cuando en una descripción formal aparece la palabra sentencias se quiere simbolizar:

$$\text{sentencia}\left[,\ \text{sentencia}\right]^{*}.$$

donde sentencia formalmente es:

$$\left\{\begin{array}{l}\text{sentencia operativa}\\ \text{sentencia de selección}\\ \text{sentencia de iteración}\end{array}\right\}$$

a) Sentencias operativas:

Las sentencias operativas son tres:

1. **TOME [UNA] [CARTA] DE [LA] PILA nombre de pila**

2. $\left\{\begin{array}{l}\textbf{DEPOSITE LA CARTA}\\ \textbf{DEPOSITELA}\end{array}\right\}$ **EN [LA] PILA nombre de pila**

3. $\left\{\begin{array}{l}\textbf{INVIERTA [LA] CARTA}\\ \textbf{INVIERTALA}\end{array}\right\}$

La ejecución por UCP de una sentencia operativa **TOME** presupone que la pila cuyo nombre figura en la sentencia la hemos definido.

La carta que figura en el tope de la pila es la que toma UCP y cualquier referencia posterior que hagamos al ente CARTA se interpretará como una referencia a la última carta tomada por UCP (siempre que no haya sido depositada).

DEPOSITE ordena a UCP dejar la carta que tiene en la mano en ese momento en una pila determinada.

La ejecución de un DEPOSITE presupone que UCP tiene, efectivamente, una carta en la mano, lo que se consigue por medio de la ejecución de una sentencia TOME.

UCP detectará un error de ejecución si no tuviera una carta en la mano y le ordenaremos que DEPOSITE. Tampoco la UCP admite la ejecución de dos TOME sin un DEPOSITE entre ellos, detectando también un error de ejecución. UCP también reconoce como error de ejecución el tratar de tomar de una pila vacía.

El verbo INVIERTA se refiere a la carta que UCP tiene en la mano. UCP reconoce dos estados de la carta: BOCA ARRIBA y BOCA ABAJO.

La ejecución por UCP de una sentencia INVIERTA altera el estado de la CARTA: si este era BOCA ABAJO pasará a ser BOCA ARRIBA y viceversa.

La sentencia INVIERTA presupone que UCP tiene una carta en la mano, de lo contrario detecta un error de ejecución.

b) Sentencia de selección:

La sentencia SI nos permite solicitarle a UCP que evalue una condición y proceda de acuerdo a ella, a la ejecución de un bloque u otro, por lo que la secuencia normal de ejecución puede depender de las condiciones halladas en las pilas, o sea, de la situación o el estado de las pilas.

Formalmente una sentencia SI se escribe:

SI condición sentencias { SINO sentencias NADA MAS | SINO NADA MAS }

Si la condición es verdadera en el momento en que UCP la analiza, entonces ejecutará el primer bloque de sentencias, o sea, las sentencias que preceden al SINO.

Si la condición fuese falsa, UCP ejecutará las sentencias que hubiera entre el SINO y el NADA MAS, y si no las hubiera, la sentencia que sigue a la sentencia SI, es decir, la que empieza después de la coma (,) que sigue a NADA MAS.

Las sentencias que podemos escribir en un SI son todas las de TIMBA. Por lo tanto, un SI puede contener otro SI (y aún un MIENTRAS). La estructura de un SI dentro de otro SI se llama "NIDOS DE SI" o "SI ANIDADOS".

Ejemplo:

```
SI LA PILA A ESTA VACIA
  TOME UNA CARTA DE PILA B
SINO
  TOME UNA CARTA DE PILA A
NADA MAS,
INVIERTALA,
DEPOSITELA EN PILA C,
```

UCP verificará si hay cartas en la pila A, y procederá, se acuerdo con ello, a tomar de la pila A o de la pila B, si la pila A estuviera vacía. En ambos casos invertirá la carta y la depositará en el tope de la pila C.

c) Sentencia de iteración:

La sentencia MIENTRAS nos permite ordenarle a UCP realizar en forma repetida y condicionada un bloque de sentencias, el cual es gobernado por una condición verificable durante la ejecución.

MIENTRAS condición sentencias REPITA

Cuando UCP encuentra durante la ejecución de un programa una sentencia MIENTRAS, evalúa primero la condición, y si ésta es verdadera, ejecuta las sentencias que siguen hasta el REPITA, vuelve a evaluar la condición, volviendo a ejecutar todas las sentencias si la condición es aún verdadera, y así, hasta que la condición sea falsa. Cuando la condición es falsa, ya sea en la primera, segunda o enésima vez, UCP saltea la ejecución de las sentencias que siguen hasta el REPITA, prosiguiendo con las sentencias que siguen al MIENTRAS, esto es, la que comienza después de la coma (,) que sigue al REPITA.

Ejemplo:

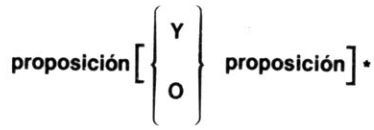
```
MIENTRAS LA PILA B NO ESTA VACIA
  TOME DE LA PILA B,
  DEPOSITELA EN LA PILA C
REPITA,
TOME DE LA PILA D.
```

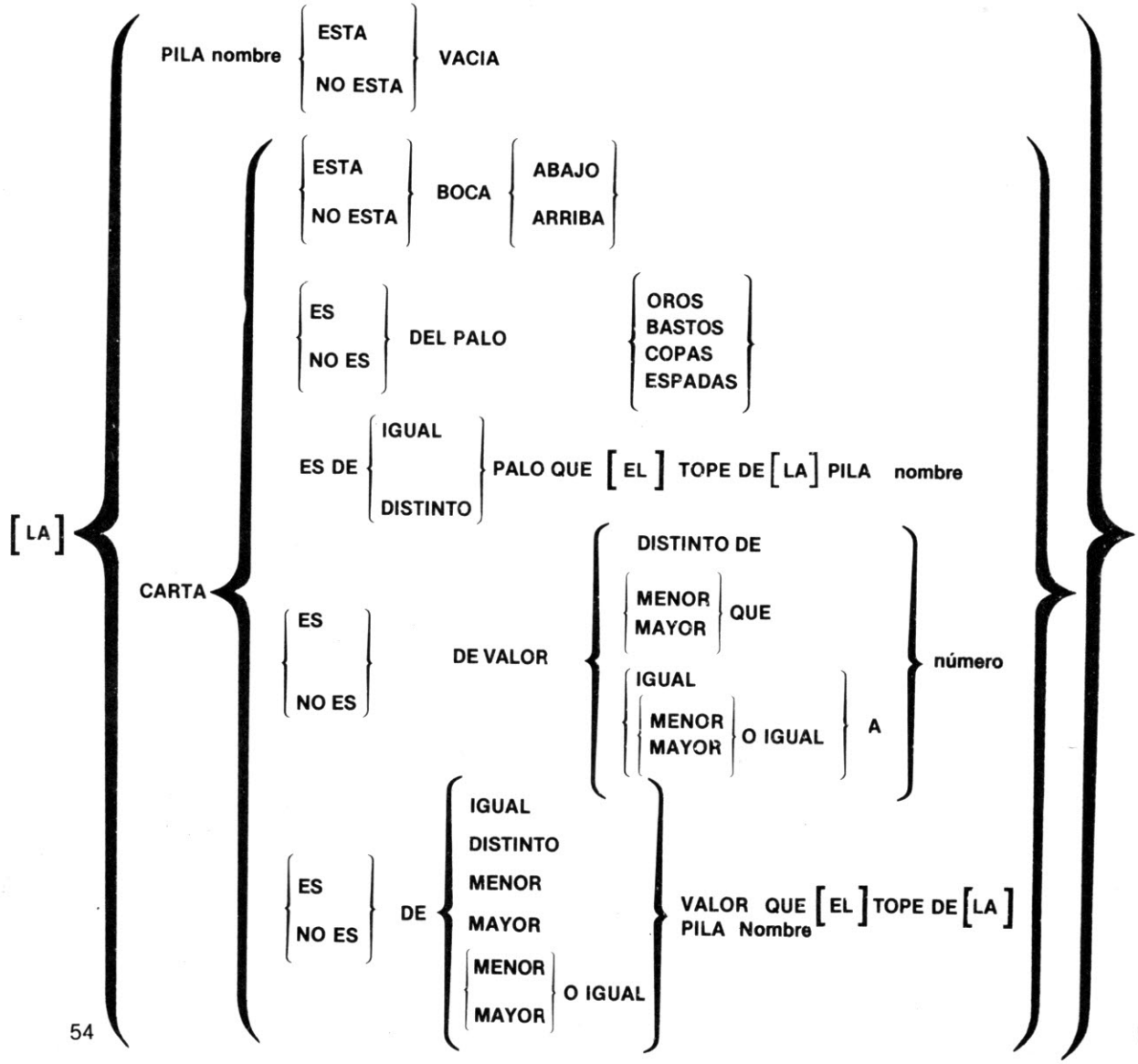
UCP verificará si la pila B no está vacía, si no lo está pasará las cartas de la pila B, de a una, a la pila C. Una vez vacía la pila B, UCP tomará la carta del tope de la pila D.

5. Condición

Una condición en TIMBA es una proposición lógica, simple o compuesta, que emite un juicio sobre el estado de las pilas o la carta y que puede ser evaluado como "verdadero" o "falso".

Formalmente, una condición de TIMBA es:





a) Pila vacía:

Formalmente la condición de pila vacía se escribe:

[LA] PILA nombre { ESTA / NO ESTA } VACIA.

UCP reconoce si una pila está vacía cuando no tiene cartas en la pila. Si hubiera una, dos o más cartas en la pila, PILA nombre ESTA VACIA sería reconocido como "falso" por UCP mientras que PILA nombre NO ESTA VACIA sería "verdadero". Como UCP reconoce un error de ejecución si se intenta tomar cartas de una pila vacía, la condición de pila vacía nos permite construir sentencias que eviten tomar de pilas sin cartas.

b) Carta boca abajo

Formalmente una condición de carta boca abajo o boca arriba se escribe:

[LA] CARTA { ESTA / NO ESTA } BOCA { ABAJO / ARRIBA }

UCP reconoce dos estados en la carta: BOCA ABAJO y BOCA ARRIBA. Si la carta estuviera BOCA ABAJO, UCP no puede hacer comparaciones, por lo que esta condición nos permite construir sentencias que la inviertan condicionalmente.

Si no hubiera CARTA definida para UCP, esto es, si ninguna carta hubiera sido tomada, UCP reconoce un error de ejecución.

c) Palo de carta

Una carta de TIMBA es un elemento biestable con dos atributos: palo y valor. Palo de una carta es uno de los cuatro palos de la baraja española: OROS, BASTOS, COPAS o ESPADAS. Valor es un número del 1 al 7 o del 10 al 12.

[LA] CARTA { { ES / NO ES } DEL PALO { OROS / BASTOS / COPAS / ESPADAS } ; ES DE { IGUAL / DISTINTO } PALO QUE [EL] TOPE DE [LA] PILA nombre }

En la primera de las dos opciones, UCP compara el palo de la CARTA contra uno de los cuatro palos de la baraja española: OROS, BASTOS, COPAS, ESPADAS, detectando valor "verdadero" si la condición la escribimos con NO ES y los palos no coinciden o con ES y los palos coinciden; y "falso" en el caso inverso: la escribimos con ES y los palos son distintos o la escribimos con NO ES y los palos coinciden.

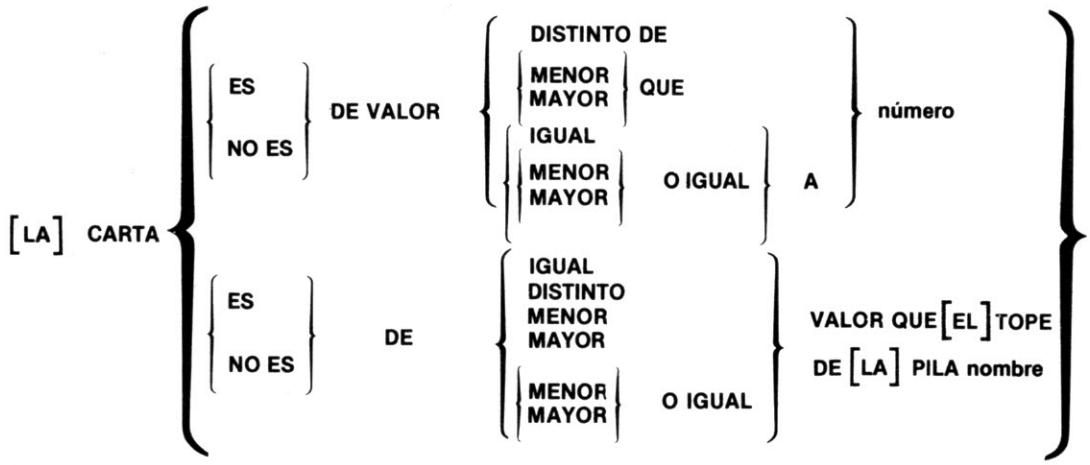
UCP detecta un error de ejecución si no tiene carta o si ésta estuviera BOCA ABAJO.

En la segunda opción, UCP compara el palo de la carta contra el palo de la carta que está en el tope de la pila "nombre", detectando valor "verdadero" si la condición la escribimos con IGUAL y los palos coinciden o si la escribimos con DISTINTO y los palos no coinciden, siendo para UCP "falsa" la condición si la escribimos con IGUAL y los palos no coinciden o si la escribimos con DISTINTO y el palo es el mismo.

UCP detecta error de ejecución si no tiene carta en la mano, o si cualquiera de las dos está BOCA ABAJO, o si la pila nombrada ESTA VACIA.

d) Valor de carta

Formalmente, la condición de valor de carta se escribe:



En la primera de las dos opciones UCP compara el valor (1 al 7 ó 10 al 12) de la carta contra el "número".

La comparación se realiza de acuerdo con las reglas de los números naturales, y cada uno de los operadores relacionales (DISTINTO DE, MENOR QUE, MAYOR QUE, IGUAL A, MENOR O IGUAL A y MAYOR O IGUAL A) representan exactamente lo que significan en el lenguaje diario.

UCP reconoce la proposición como "verdadera" si la hemos escrito con la opción ES y la relación se verifica o si la hemos escrito con NO ES y la relación no se verifica.

UCP reconoce un error de ejecución si no tiene carta en la mano o si está BOCA ABAJO.

En la segunda opción, UCP compara el valor de la carta, en el sentido de los números naturales, contra la carta que está en el tope de la pila nombrada.

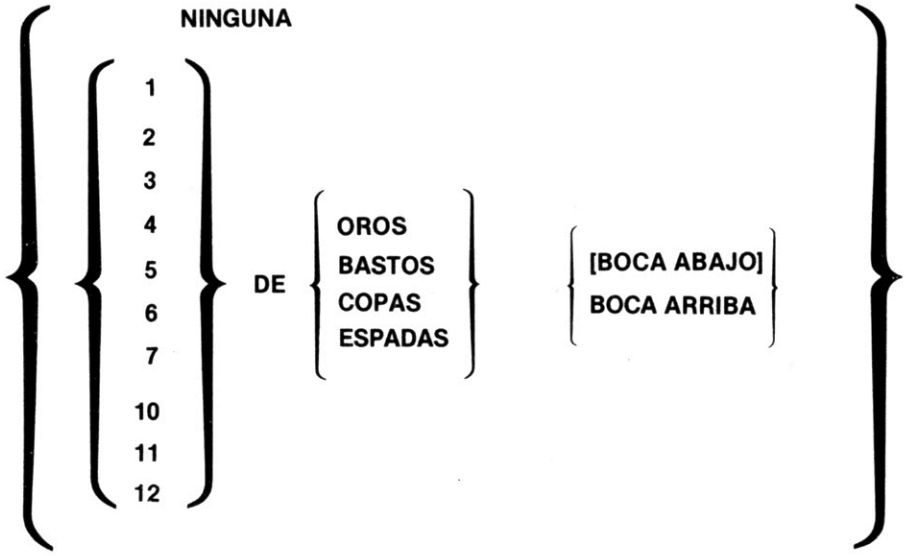
UCP reconoce la proposición como "verdadera" si la hemos escrito con la opción ES y la condición se verifica o si la hemos escrito con NO ES y la relación no se verifica.

UCP reconoce un error de ejecución si no tiene carta en la mano, si la PILA ESTA VACIA o si por lo menos una de las dos cartas de la comparación está BOCA ABAJO.

6. Descripción de los datos de la pila

Cuando queremos ejecutar un programa, UCP nos pide que le describamos las cartas que tiene cada una de las pilas.

Formalmente, la descripción de los datos de la pila se escribe:



Todas las cartas las describimos por su número y palo y suponemos que están BOCA ABAJO.

Si deseamos invertir la carta al definir la PILA, esto es, si deseamos que la carta aparezca BOCA ARRIBA, después de la descripción de la carta debemos escribir BOCA ARRIBA.

7. Palabras reservadas de TIMBA

La siguiente es una lista de palabras reservadas de TIMBA. Estas palabras no pueden ser usadas como nombres de pila.

ABAJO
ARRIBA
BASTOS
BOCA
CARTA
COPAS
DE
DEL
DEPOSITE
DEPOSITELA
DISTINTO
EL
EN
ES
ESPADAS
ESTA
IGUAL
INVIERTA
INVIERTALA
LA
MAS
MAYOR
MENOR
MIENTRAS
NADA
NINGUNA
NO
O
OROS
PALO
PILA
QUE
REPITA
SI
SINO
TOME
TOPE
UNA
VACIA
VALOR
Y
1
2
3
4
5
6
7
10
11
12

8. Mensajes de error de ejecución

Son aquellos que emite UCP cuando detecta un error de ejecución:

- ME PIDE UD. QUE TOME DE LA PILA nombre QUE ESTA VACIA!
- ME PIDE UD. QUE TOME DE LA PILA nombre Y YO YA TENGO UNA CARTA: número DE palo
- ME PIDE UD. QUE INVIERTA LA CARTA, PERO YO NO TENGO CARTA.
- ME PIDE UD. QUE DEPOSITE EN LA PILA nombre Y YO NO TENGO CARTA!
- EN LA LINEA nro de línea UD ME PIDE QUE MIRE LA CARTA EN MI MANO. Y YO NO TENGO NINGUNA!
- COMO HAGO PARA COMPARAR? LA CARTA DE MI MANO ESTA BOCA ABAJO!
- COMO? EL TOPE DE LA PILA nombre ESTA BOCA ABAJO. COMO COMPARO YO?
- COMO? LA PILA nombre ESTA VACIA! QUE COMPARO YO?

Ejercicios

Cuando no aclara el estado de la carta, se supone que está boca arriba.

1 Se tienen dos pilas no vacías de cartas, A y B. Generen otra pila C:
a) donde se encuentren solamente los topes de A y B.
b) Idem pero invertidos.

2 Dada una pila con al menos dos cartas, escriban un programa TIMBA que:
a) Intercambie las cartas que ocupan el primero y segundo lugar en la pila, o sea, el tope pasa a ser la segunda y la segunda pasa a ser el tope.
b) Idem pero dejando el nuevo tope boca abajo.

3 Dada una pila A no vacía de cartas boca arriba y/o boca abajo, invertir el tope de la misma. Si éste está boca abajo (dejar todas las cartas en A).

4 Sacar de una pila, con al menos dos cartas, la segunda si ésta es de oros.

5 Dada una pila con al menos dos cartas, si el tope es el rey de oros extraer la segunda, en caso contrario invertirla, dejándola en el mismo lugar.

6 Dadas 3 pilas no vacías A, B y C si los 3 topes son del mismo valor, extraer las cartas que ocupan el segundo lugar y generar otra pila "SEG" con las mismas, las pilas no necesariamente tienen al menos dos cartas.

7 Dadas 3 pilas de cartas A, B y C, generar otra pila D, donde la pila B esté sobre la A y la C debajo de la A.
a) No se exige que se mantenga el orden en las pilas, no usar pilas auxiliares.
b) Manteniendo el orden de las pilas, usando solamente una pila auxiliar.

8 Dada una pila A de cartas, generar otra pila donde se encuentren solamente los bastos mayores que tres.

9 Seleccionar una pila no vacía, aquellas que sean copas generando una pila con aquellas, dejar la pila A con las cartas restantes como estaban. Una vez ejecutado el programa, ¿qué circunstancia le indica que en la pila primitiva no había ninguna del palo copas?

10 Idem al ejercicio anterior, pero las cartas pueden estar boca arriba y/o boca abajo, no interesa el estado final de las cartas.

11 Dada una pila B con al menos dos cartas, extraer el tope y la base de la misma y generar una pila con ellas, las demás cartas deben quedar en el mismo orden en la pila B.

a) Podría servir el algoritmo diseñado para saber si la pila B tiene exactamente 2 cartas?
b) ¿Qué circunstancia le indica, luego de ejecutado el programa, este hecho?

12 Dada una pila A, no vacía, de cartas, generar cuatro pilas con sólo ases, reyes, sotas y caballos, las restantes cartas deben quedar en la pila A.

13 Dadas dos pilas de cartas boca arriba y/o boca abajo, intercalar las dos pilas dejando la carta en estado contrario al inicial, la que estaba boca abajo tiene que quedar boca arriba y viceversa.
Observación: debe intercalarse el mayor número posible de cartas.

14 Dada una pila A de cartas de la que se sabe que tiene dos ases, extraer de la misma los ases y todas las cartas intermedias. Las restantes cartas deben quedar en la pila A.

15 Seleccionar de una pila A las cartas que anteceden a una espada.

16 Seleccionar de una pila A las cartas que siendo de espadas anteceden a un as.

17 Dada una pila A, determinar (dejando una señal), si el número de sus elementos es par o impar.
La señal deberá ser una carta invertida.

18 Dada una pila A, separar las cartas que coincide en palo con la que sigue.

19 Idem al anterior pero con la que antecede.

En la Unidad III hemos planteado las etapas fundamentales del proceso del Pensar: detectar un problema, plantear un algoritmo, describir un programa, comunicarnos con un lenguaje compartido.

La Unidad IV nos permitió comprender la interrelación que existe entre el Pensar y el Hacer, conocer la estructura de un lenguaje de programación, y desarrollar programas para resolver problemas.

En la Unidad II abordamos el estudio de la estructura y función de la Computadora.

En esta Unidad se destaca claramente la función del Hombre en relación con la Computadora, a quien no sólo debe manejar, sino especialmente darle órdenes. Para ello debe resolver el Problema detectado.

Aparece así la figura del Programador, y su rol profesional.

En esta Unidad se dedica especial atención a la Programación Estructurada que ustedes necesitan conocer para diseñar programas en Computadoras cada vez más perfectas y menos costosas.

En la lectura y análisis del texto de esta Unidad, y su posterior discusión, en reuniones de dos horas cada una, en grupo, con la conducción de su coordinador, esperamos que Uds. logren cumplir los siguientes:

• Objetivos Generales

1. Conozcan las funciones de un Programador.
2. Comprendan la relación que existe entre la Máquina, el Programa y el Lenguaje.
3. Conozcan el diseño de la Programación Estructurada.
4. Comprendan que el estudio de la fundamentación teórica de una actividad la facilita.
5. Valoren el rol del Programador.

• Objetivos Específicos

1.1. Mencionen dos funciones del Programador.
1.2. Describan cada una de las funciones mencionadas.
2.1. Señalen las semejanzas y diferencias entre la algoritmización del problema detectado y su expresión en términos ejecutables por la máquina.
3.1. Redefinan un programa en función de la posible actividad de la máquina.
3.2. Informen acerca de las exigencias que requiere el proceso correspondiente a la solución del problema detectado, contando con la ayuda de la Computadora.
3.3. Describan la técnica de Estructuración Jerárquica.
3.4. Diagramen estructuras de datos propuestos por el grupo.

3.5. Describan el proceso de diseño de un Programa.

4.1. Comparen la posibilidad actual de realizar un ejercicio de programación, en relación a la correspondiente, al solucionar los de la Unidad anterior.

5.1. Participen activamente en el grupo para diseñar algún Programa sencillo.

Introducción

La historia de la tecnología nos muestra la tendencia del hombre a generar el máximo de trabajo con el mínimo de esfuerzo. Cosa que va logrando con el recurso de su ingenio, cuya expresión más rica es, quizás, el instrumento.

El instrumento facilita las tareas del hombre y, en ciertos casos, llega a reemplazarlo.

Por ejemplo: con el invento del remo, el hombre multiplica su fuerza muscular y aumenta sus posibilidades de navegación. Luego, con la vela, introduce un nuevo instrumento que le permite liberarse de la dura tarea de remar. Finalmente, con el motor, rompe su dependencia del clima y de los vientos, satisfaciendo mejor esa tendencia de que "algún otro" haga las cosas por él, y mejor.

Esta misma tendencia quedó demostrada en nuestra historia de la computación. El paso del contador de arena al ábaco significó facilitar enormemente las tareas de cálculo aritmético; con la calculadora de Pascal el hombre es directamente reemplazado en las tareas de sumar y restar, y luego, con el contador de Leibnitz, en las tareas de multiplicar, dividir y extraer raíces. En 1890, la máquina de Hollerith ejecutó las tareas de leer, contar y clasificar las tarjetas de los censados. En fin, la ENIAC (1946), hizo en dos horas un trabajo de cálculo equivalente al de 100 ingenieros durante un año.

Sin duda el desarrollo tecnológico desplegado en los últimos 50 años no tiene parangón en toda la historia de la humanidad, siendo la computación uno de sus exponentes más avanzados.

Es que la computadora, como ningún otro instrumento, presenta posibilidades de reemplazar al hombre en la ejecución de tareas, permitiéndole ahorrar y disponer no ya de su energía física sino mental. Dicho en pocas palabras: con la ayuda de otros instrumentos, el hombre ejecuta sus tareas; con la computadora, da órdenes para su ejecución.

Es por eso que el objetivo no es ya saber cómo manejar una máquina, sino cómo darle órdenes.

1. La Tarea, el Instrumento, el Programador

Es necesario resolver un determinado problema, surgido de las circunstancias reales de la vida (recordemos el ejemplo de la goma pinchada del automóvil).

La resolución del problema impone la ejecución de una tarea (la tarea de cambiar la goma pinchada por la de repuesto). Para ejecutar esta tarea se disponen de determinados recursos instrumentales (un criquet y una llave de cruz).

Sólo falta entonces una persona capaz de ejecutar todas las operaciones, y de una sucesión correcta de pasos, para transitar de la situación A (coche con la goma pinchada) a la situación B (coche con la goma cambiada). (Unidad III). Esta persona es el recurso humano.

La computadora es un recurso instrumental que, a diferencia del criquet y de la llave de cruz, reemplaza al hombre en la ejecución de la tarea.

Este recurso instrumental, que es la computadora, exige un recurso humano especial, al que llamamos programador.

El programador es un profesional que tiene la capacidad de

- definir una tarea para la computadora, desde el principio al fin, con la correcta secuencia de pasos,
- expresar esta tarea en términos de operaciones simples, y por lo tanto, ejecutables por la computadora.

Veamos cada una de estas funciones.

La definición de la tarea no es otra cosa que la formulación del algoritmo. Esto exige al programador la capacidad de identificar correctamente cual es el problema, cual es el camino que conduce a su resolución y cuales los pasos o etapas del mismo, puesto que toda resolución es un proceso.

El algoritmo nos demuestra si el problema tiene o no solución y que ésta, de existir, pueda darse en un proceso que incluye tales y cuales pasos.

He aquí, entonces, la primera función del programador: la "algoritmización" del problema, tarea que realiza describiendo los pasos en su propio idioma, como ya se dijo.

La otra función del programador es la formulación de la tarea, trazada en el algoritmo, en un lenguaje de programación.

En otras palabras, el programador traduce la tarea en sentencias ejecutables para la máquina y en un lenguaje tolerable para ésta.

Estas sentencias de ejecución, formuladas en un lenguaje de programación, son operaciones simples que la máquina puede ejecutar. En la experiencia de TIMBA, por ejemplo, vimos que la UCP reconoce solo tres tipos de operaciones a ejecutar: TOME, DEPOSITE, INVIERTA.

Entonces, sintetizando, el programador tiene dos funciones: algoritmizar el problema y formular la tarea en términos de sentencias ejecutables.

El algoritmo se expresa en el lenguaje usual del programador, mientras que las sentencias ejecutables (es decir el programa), se expresan en lenguaje de programación.

El lenguaje de programación, como ya vimos, puede ser de alto nivel, de bajo nivel, o directamente lenguaje de máquina.

La computadora sólo es activada por el lenguaje de máquina, basado en el sistema binario. El lenguaje de bajo nivel es traducido al lenguaje de máquina por los ensambladores, mientras que los lenguajes de alto nivel son traducidos a lenguajes de máquina, gracias a los intérpretes o compiladores.

TIMBA y otros lenguajes que estudiaremos son de alto nivel, puesto que se asemejan más a nuestro lenguaje usual.

2. El Perfil del Programador

La tecnología de la máquina computadora (nos referimos a la parte "hardware", a la "ferretería") ostenta un perfeccionamiento acelerado y prodigioso en las últimas décadas. Microcircuitos de alta complejidad dan lugar a máquinas pequeñas, compactas, de gran precisión. Los sistemas de entrada y de salida son cada vez más funcionales y perfectos. Y las memorias auxiliares han ampliado las posibilidades de almacenamiento de datos y programas, permitiendo desarrollar tareas de gran complejidad.

Pero otros son los derroteros por donde ha debido avanzar la parte "software",es decir, el sostén lógico de la computadora. Esta parte de la historia involucra a los programas y sus artífices: los programadores. Es una historia lenta, de tramos penosos y llenos de vericuetos.

Muy brevemente diremos que, en la década que va del año 40 al 50, los programadores eran muy pocos. Esas grandes moles que fueron las primeras computadoras no se movían de los lugares donde se construyeron, de suerte que su utilización era poco universal, para trabajos muy específicos y dentro de un círculo muy pequeño de personas.

La tarea de hacer programas, en esas circunstancias, no era considerada como algo propio de una profesión. Un buen programador, se pensaba entonces, debía ser aficionado a los rompecabezas y a los juegos de tretas inteligentes, para optimizar los procesos.

En las décadas siguientes aparecen máquinas más poderosas, que plantean difíciles problemas para su plena utilización. Se desata una crisis entre los programadores. Pero el poder de las máquinas sigue creciendo, y con él la ambición de la sociedad para utilizarlas.

Una primera respuesta para intentar reducir la complejidad del trabajo fue el desarrollo de los lenguajes de alto nivel. Esto significó la introducción de técnicas de codificación más satisfactorias, pero con muy pocas ayudas efectivas para la concepción del algoritmo.

¿Cuál era el problema para ese entonces? La producción de grandes sistemas, donde mantener el crecimiento de los programas, dentro de un esquema comprensible, era un gran desafío.

Mientras la producción y el perfeccionamiento de máquinas computadoras se hacían menos costosos, y los precios de oferta en el mercado iban disminuyendo ostensiblemente, los precios de la "mano de obra" (las horas hombre del programador) aumentaban.

Esta situación intensificó la necesidad de aprender a programar de una manera efectiva, hasta producir programas intelectualmente manejables.

En nuestros días, el programador es un profesional reconocido socialmente y dotado de una disciplina que se fundamenta en conocimientos sólidos y coherentes.

3. La programación estructurada

Veremos cómo diseñar programas estructurados.

Este tipo de programación ha sido muy aceptada y tiene la virtud de ponernos a salvo de muchos defectos, que entorpecen el curso de la tarea.

a) Desde la tarea a realizarse (problema) hasta la instrucción ejecutable simple (sentencia).

Decíamos que, si la computadora es una máquina que nos reemplaza en la ejecución de una tarea, entonces debemos aprender a darle órdenes. El programa es un conjunto de órdenes, que activan a la máquina.

Dar órdenes para lograr que una tarea sea ejecutada, exige expresarse en un lenguaje adecuado, de suerte que el ejecutor sepa exactamente qué tiene que hacer.

Esto de dar correctamente una sucesión de órdenes, para desarrollar una tarea más o menos compleja, lo podemos entender con un ejemplo de la vida real.

Supongamos que se produce un accidente de tránsito y quedan varios heridos. Junto al lugar pasa una persona que sabe de primeros auxilios, a quien llamaremos, en adelante, "el socorrista". Pero ocurre que nuestro buen socorrista está físicamente impedido de ejercer su función de tal, y pide que alguien ejecute sus instrucciones. A este llamaremos "el ejecutor". El socorrista, en medio del descalabro, impuso la calma, envió a llamar la ambulancia y despejó a los curiosos. Entre los heridos, pone su atención en una persona que permanece inconsciente. A ésta, la llamaremos "la víctima".

La tarea planteada al socorrista, en estas circunstancias, es **mantener viva a la víctima hasta que llegue el auxilio médico.**

Ahora bien, la tarea de "mantener viva a la víctima", exige una "**sub-tarea**": verificar si tiene un paro respiratorio. Esto puede acarrear la muerte en menos de un minuto.

A continuación vamos a desarrollar un diálogo imaginario entre el socorrista (S) y el ejecutor (E).

S: Verifique si hay paro respiratorio

E: No sé
S: Ponga a la víctima en posición correcta
E: No sé
S: Ponga a la víctima boca arriba
E: Sí, señor
S: Arrodíllese junto al hombro derecho
E: Sí, señor

S: Ponga su mano derecha bajo el cuello
E: Sí, señor
S: Ponga su mano izquierda sobre la frente
E: Sí, señor
S: Empuje la mano derecha hacia arriba y la izquierda hacia abajo, haciendo girar la cabeza
E: Sí, señor

S: Establezca la falta de respiración
E: No sé
S: Acerque su oido izquierdo junto a la boca
E: Sí, señor
S: Mire el pecho de la víctima
E: Sí, señor

A continuación el socorrista le pregunta al ejecutor, si siente salida de aire de la boca de la víctima (por el sonido y el calor) y si el pecho se mueve.

Llamaremos T1 a la tarea de "verificar paro respiratorio".

El socorrista, primero, transformó toda esta tarea, en una orden: "verifique si..." Pero como el ejecutor no podía obedecer, descompuso T1 en:

T1-1: poner a la víctima en posición correcta, y

T1-2: establecer la falta de respiración.

Nuevamente el socorrista da una orden compleja: "ponga a la víctima en posición correcta" con igual resultado negativo por parte del ejecutor.

Entonces, el socorrista desglosa la orden en cinco órdenes simples que el ejecutor las entiende y transforma en acciones, una a continuación de la otra, hasta cumplir con la tarea de: "colocar a la víctima en posición correcta"

También el socorrista transformó la tarea de "establecer falta de respiración" en una sola orden, la que debió descomponer en dos órdenes simples, comprensibles para el ejecutor.

En el desarrollo de este ejemplo hay, sin duda, algo ficticio, y es que el socorrista no le pregunta al ejecutor qué sabe de Primeros Auxilios, si no que lo va evaluando en la práctica, lo que constituye una pérdida de tiempo. Si el ejecutor informara al socorrista que es un experto en el tema, bastaría una sola orden: "mantenga viva a la víctima hasta que"... Si fuera un estudiante avanzado, le ordenaría, en una primera instancia: "verifique si hay paro respiratorio", y esperaría su información. Si fuera un estudiante medio, entonces descendería a un tercer nivel de concreción y le ordenaría, primero, que ponga a su víctima en posición correcta y luego que establezca la falta de respiración, como únicas órdenes. Por último, si el ejecutor dice ser un lego en la materia, el socorrista debiera, sin pérdida de tiempo, adoptar un nivel de órdenes simples.

Sin embargo vamos a continuar con esta ficción de "tantear" la comprensión del ejecutor, impartiéndole primero una orden compleja, para descomponerla en otras menos complejas y, por último, en varias órdenes simples.

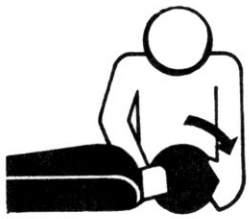
Quedamos en que el ejecutor está en condiciones de verificar si hay falta de respiración". Esta verificación la realiza por acciones simples: sentir calor sobre la oreja, escuchar salida de aire, ver movimientos del pecho. Si el ejecutor no registra ninguna de estas sensaciones, el socorrista diagnostica paro respiratorio.

Supongamos que hay paro respiratorio.

Frente a este problema, la tarea es: aplicar respiración artificial.

El diálogo entre el socorrista y el ejecutor, podría sucederse así:

S: Aplique respiración artifical



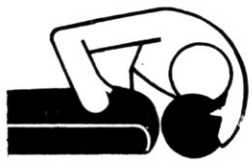
E: No sé
S: Despeje vías respiratorias
E: No sé
S: Mantenga la cabeza de la víctima bien inclinada hacia atrás
E: Sí, señor
S: Inmovilice la cabeza, presionando con la mano izquierda sobre la frente de la víctima
E: Sí, señor

S: Saque su mano derecha del cuello y con ella, ábrale bien la boca
E: Sí, señor

S: Introduzca su dedo índice en la boca
E: Sí, señor
S: Recorra la boca con el dedo en busca de algún cuerpo extraño. Si hay, sáquelo. Si no hay, ejecute mi próxima orden
E: Sí, señor

S: Inyéctele aire a la víctima
E: No sé
S: Mantenga abierta la boca de la víctima
E: Sí, señor
S: Con el índice y pulgar de su mano apoyada en la frente, tape la nariz
E: Sí, señor

S: Llene de aire sus propios pulmones
E: sí, señor
S: Abra su boca y cubra con sus labios los de la víctima
E: .....
S: Sople
E: .....
S: Ponga su mano derecha sobre el pecho de la víctima y verifique si el pecho se levanta
E: .....



S: Si no se levanta, verifique que la cabeza de la víctima esté bien inclinada hacia atrás. Si se levanta continúe
E: Sí, señor
S: Mientras la víctima no respire por sí misma, tome aire, cubra con sus labios los de ella, sople. Repita
E: Sí, señor

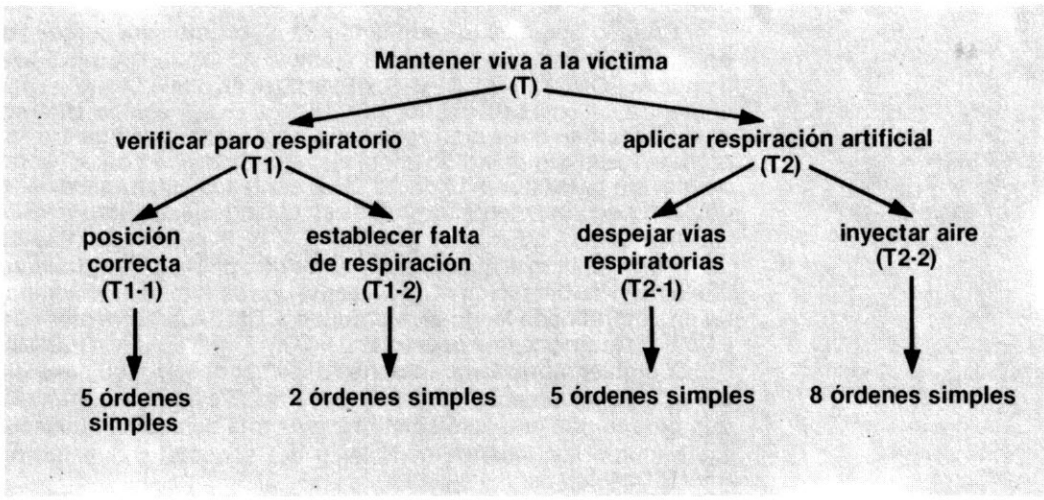
El socorrista, como puede apreciarse, descompuso la tarea "aplicar respiración artificial" (T2) en:

T2-1 "Despejar vías respiratorias", la que a su vez descompuso en 5 órdenes simples.

T2-2 "Inyectarle aire a la víctima", la que fue descompuesta en ocho instrucciones simples.

Recapitulando:

Ante el hecho de una víctima inconciente, el socorrista se plantéó una tarea T: mantenerla viva, hasta que llegue la ambulancia. Sus conocimientos de como socorrer le indican que lo primero es verificar si la víctima respira o no, tarea esta que denominamos T1. Como la víctima no respiraba se abrió otra tarea, la T2: aplicar respiración artificial. Pero recordemos que el socorrista no atendió a la víctima, sino un ejecutor, profano en el tema. Esto obligó a ir reformulando sus órdenes en niveles cada vez más simples, hasta referirse a acciones directas y concretas, no factibles de ser más simplificadas.



Así tenemos que una tarea se ha ido descomponiendo, nivel por nivel, en dos subtareas, luego de cuatro tareas menores y, finalmente, en 20 órdenes simples, traducibles a maniobras directas y concretas.

Toda esta estructura constituye el algoritmo del socorrista, expresado en su propio idioma, que es también el idioma del ejecutor.

Si el socorrista debe asistir a una víctima en alguna calle de Nueva York; si no habla otro idioma que el castellano y si el ejecutor no entiende otro idioma que el inglés, entonces habrá que recurrir a un intérprete.

En esta nueva situación las órdenes del socorrista, al ser traducidas al inglés, se harían comprensibles al ejecutor.

La obra del socorrista sería el algoritmo; la obra del traductor sería el programa y la obra del ejecutor, el proceso de devolverle la respiración a la víctima.

Esta situación de socorrista-traductor-ejecutor es la que más se asemeja a la del programador, que debe traducir su algoritmo (o hacerlo traducir por otro) a un determinado lenguaje, tolerable para una determinada máquina.

En síntesis, para impartir correctamente órdenes a un ejecutor, debemos:

1° - hablarle en su idioma

2° - hablarle en términos ejecutables

Tomando nuestra experiencia con TIMBA, este constituye un lenguaje, comprensible para determinadas máquinas.

Cuando estamos desarrollando un algoritmo para programar en TIMBA, podemos escribir, por ejemplo: "ADQUIERA una carta de pila A, PONGALA AL REVES, UBIQUELA en pila B". Pero si esta sentencia, tal cual está escrita, la entramos en la máquina, UCP no la va a ejecutar. Sería algo semejante a pretender que nuestro socorrista fuese comprendido por el ejecutor de Nueva York, directamente, sin pasar por el traductor. Esa sentencia algorítmica tiene que estar rigurosamente formulada en el lenguaje de TIMBA: "TOME UNA CARTA DE PILA A, INVIERTALA, DEPOSITELA EN PILA B"

Pero también el que realiza el algoritmo debe tomar en cuenta cuales son las operaciones que ejecuta una determinada máquina, en un determinado lenguaje. Volviendo a TIMBA, recordemos que UCP solo reconoce tres operaciones: TOME - INVIERTA - DEPOSITE. Cualquier movimiento de cartas debe ser desplegado pasando por estas tres operaciones, y sólo por ellas. Y cualquier sentencia que ordene una operación distinta sería una sentencia no ejecutable (como las órdenes complejas del socorrista, que fueron impracticables para el ejecutor).

b) La estructuración jerárquica

Mientras desarrollábamos el ejemplo del socorrista y el ejecutor señalamos que había un elemento ficticio, y es que aquel iba evaluando los conocimientos de éste, con la consiguiente pérdida de tiempo.

Si el ejecutor era un lego había que dictar solamente órdenes simples, o sea, directamente traducibles a la acción.

En la jerga de los programadores, no se habla de "ordenes simples", sino de **sentencias ejecutables**; ni tampoco de "acción", sino de **operaciones primitivas**.

Aplicándolo a nuestro ejemplo el socorrista, ante un ejecutor lego, debe pronunciar directamente sentencias ejecutables, convertibles por éste a operaciones primitivas (abra la boca de la víctima, tome aire, sople...).

En una tarea como ésta, que se desarrolló, según puede comprobarse, en 20 operaciones primitivas, es posible expresarse directamente, en 20 sentencias ejecutables. Es un proceso bien abarcable por la mente del socorrista.

Pero supongamos que, para desarrollar una determinada tarea, es necesario ejecutar 750 operaciones primitivas. Un ejemplo para nada exagerado. Imposible que un programador escriba, directamente, 750 sentencias ejecutables. Y si alguno intenta hacerlo corre el riesgo de advertir, supongamos, mientras redacta la sentencia N° 500, que ha cometido un error en la N° 14... y tendrá que desandar un larguísimo camino.

En casos como estos, que son los reales, el programador necesita dedicar su atención a una pequeña parte del trabajo, en un determinado momento. Luego pasar a otra parte, y así sucesivamente, por partes, sin perder el hilo conductor del trabajo total, ni la conexión entre sí de las partes.

Necesitamos crear un programa P, que al ejecutarse, realice la tarea T.

Descomponemos T en sub-tareas: T1, T2 y T3. Hacemos una descomposición similar de P: P1, P2 y P3. Al ejecutarse P1, es realizada T1, y así sucesivamente.

Ahora tenemos que crear cuatro componentes: P1, P2, P3 y P mismo. P es necesario para crear la correcta relación entre P1, P2 y P3.

Pero P1, P2 y P3 son, probablemente, muy complejos para ser escritos directamente en sentencias ejecutables, lo que impone continuar con la descomposición. Entonces tomamos T1 y la descomponemos en T1-1, T1-2, T1-3 y T1-4. Luego otro tanto con P1.

El proceso continua, nivel por nivel, hasta que no quede ningún componente muy complejo para ser codificado directamente en lenguaje de programación, como una sentencia ejecutable simple.

Esta es la técnica de la **estructuración jerárquica**, que nos permite crear programas largos y complejos. La complejidad y el tamaño de los programas son reflejados por el número de componentes y de niveles en la jerarquía.

El proceso de descomposición de la tarea que realiza el socorrista, y que no se justifica por la simplicidad y urgencia de la misma, resulta indispensable en las programaciones largas y complejas. Mientras el socorrista puede mantener en su mente toda la estructura jerárquica de un programa, desde T hasta la última sentencia ejecutable, el programador necesita dimensionar su tarea a escala humana, e ir por partes. De ahí la importancia de la estructuración jerárquica como técnica de trabajo.

c) Los componentes

Llamamos "componentes" a los últimos niveles de descomposición de los programas en la estructuración jerárquica.

Los componentes pueden ser de dos clases:

- elementales.
- compuestos.

Los componentes elementales son sentencias simples del lenguaje de programación. No pueden dividirse, carecen de partes. En TIMBA, equivalen a las "sentencias operativas": TOME - DEPOSITE - INVIERTA.

Los componentes compuestos son combinaciones de los componentes elementales, pudiendo ser:.

- secuenciales.
- iterativos.
- selectivos.

Un componente compuesto puede estar formado por componentes elementales y otros componentes compuestos.

La **secuencia** está formada por dos o más componentes que se ejecutan por orden de aparición.

Ej.: TIMBA
— TOME UNA CARTA DE PILA A
— SI LA CARTA ES DE OROS
INVIERTALA
SI NO NADA MAS
— DEPOSITELA EN PILA B.

En el caso del socorrista, un componente secuencial es:

- Ponga la víctima boca arriba.
- Arrodíllese junto al hombro derecho.
- Ponga su mano derecha bajo el cuello.
- Ponga su mano izquierda sobre la frente.
- Empuje la mano derecha hacia arriba.
- Empuje la mano izquierda hacia abajo.
- Acerque su oído a la boca de la víctima.
- Mire el pecho.
- Sienta la respiración de la víctima.

La **iteración** está formada por una o más componentes cuya ejecución se repite cero o más veces.

Ej. TIMBA
— MIENTRAS LA PILA B NO ESTA VACIA
TOME DE PILA B
DEPOSITE EN PILA C
REPITA

Ej.: Socorrista
mientras la víctima no respire por sí misma
tome aire
selle sus labios con los suyos
sople
repita

En ambos casos, la iteración está formada por un bloque de sentencias simples. El bloque se considera como una sola parte.

La **selección** plantea optar entre distintos caminos de ejecución.

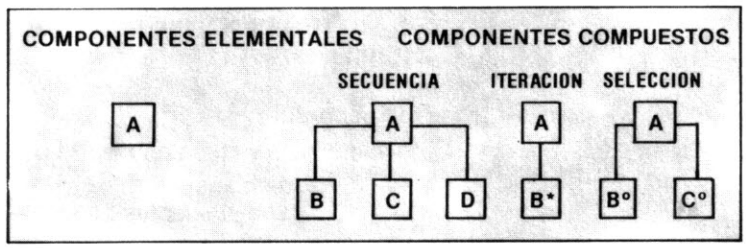
Ej.: TIMBA
— SI LA PILA A ESTA VACIA
TOME UNA DE LA PILA B
SINO
TOME UNA DE LA PILA A
SINO NADA MAS.

Ej.: Socorrista
— Si hay algún cuerpo extraño en la boca
Sáquelo
Sino
Tape la nariz
Nada más

Se ha demostrado que cualquier programa puede ser expresado como una estructura de secuencias, iteraciones y selecciones.

d) Estructura de datos y de programas

Los cuatro tipos de componentes se manifiestan tanto en la estructura de los datos como en la estructura de los programas. Más todavía, la estructura del programa debe basarse en la estructura de los datos.



*Indica repeticiones de B dentro de A

°Indica una selección (ocurre una u otra)

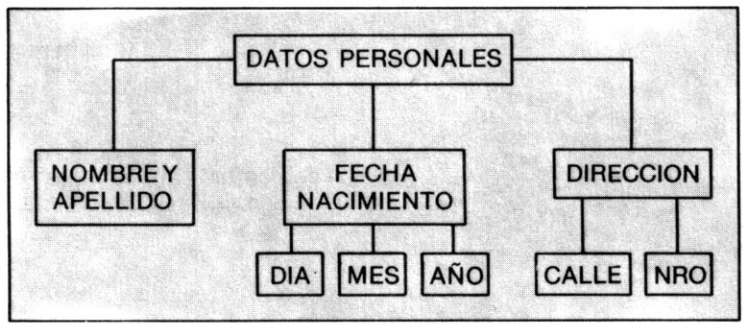
Ejemplos de Estructuras de Datos

El universo de la información está formado por datos, provenientes, todos ellos, de la realidad. Estos datos, a su vez, se ubican en determinados continentes, según sea la utilidad que deban prestar. A continuación vamos a analizar informaciones de diverso orden, cuyos datos puedan ser reconocidos como formando parte de determinadas estructuras.

DATOS PERSONALES

Nombre y apellido
Fecha de nacimiento
  Dia
  Mes } Secuencia
  Año
Dirección
  Calle } Secuencia
  Nro.

} Secuencia

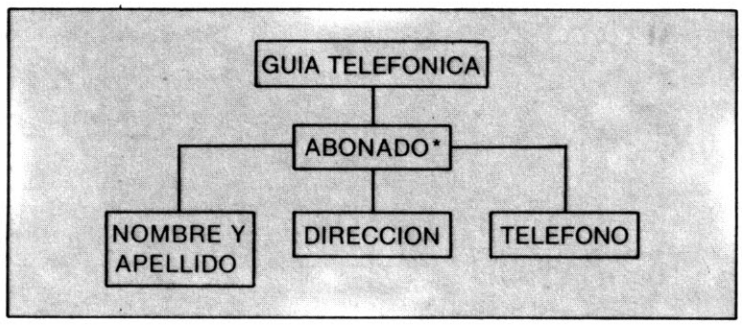


GUIA TELEFONICA

Abonado*

Nombre y Apellido
Dirección
Teléfono
] Secuencia

] Iteración

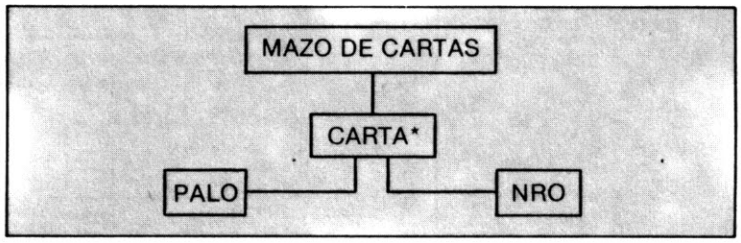


MAZO DE CARTAS

Carta*

palo
número
] Secuencia

] Iteración

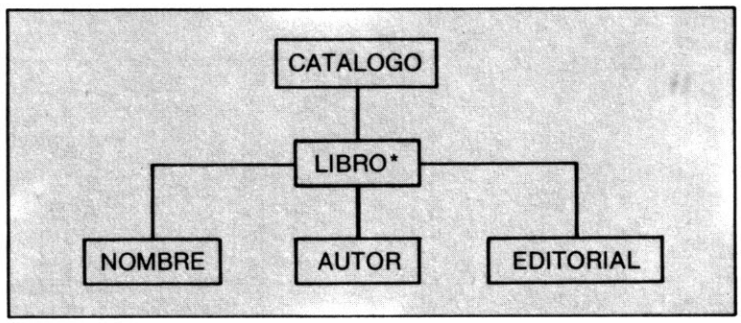


CATALOGO

Libro*

Nombre
Autor
Editorial — Secuencia

Libro* — Iteracción



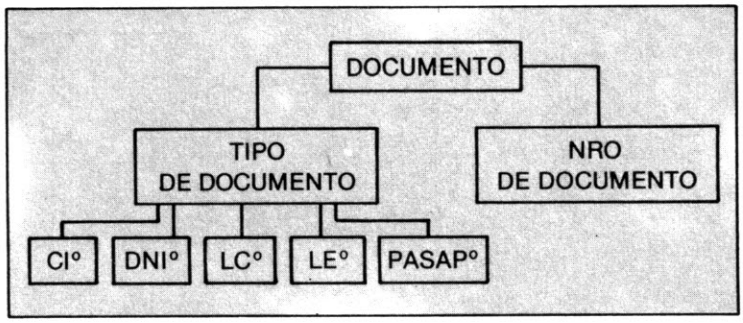
DOCUMENTO

Tipo documento
- CI°
- DNI°
- LC°
- LE°
- PASAP°

(Selección)

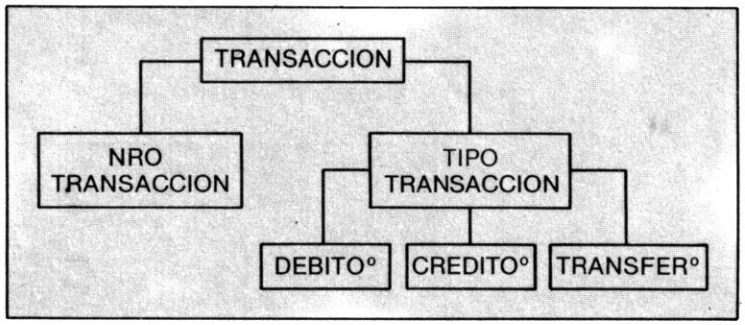
Nro documento

(Secuencia)



TRANSACCION

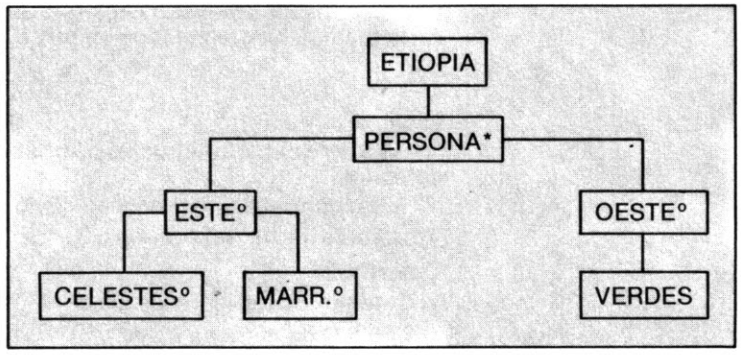
Número de transacción
Tipo transacción
Débito°
Crédito°
Transferencia°
Selección
Secuencia



Etiopía está dividida en dos partes: Este y Oeste
Toda la gente que vive en el Oeste tiene ojos verdes, pero la que vive en el Este puede tener ojos celestes o marrones.

ETIOPIA

persona*
Este°
ojos celestes°
ojos marrones°
Selección
Oeste°
ojos verdes
Selección
Iteracción



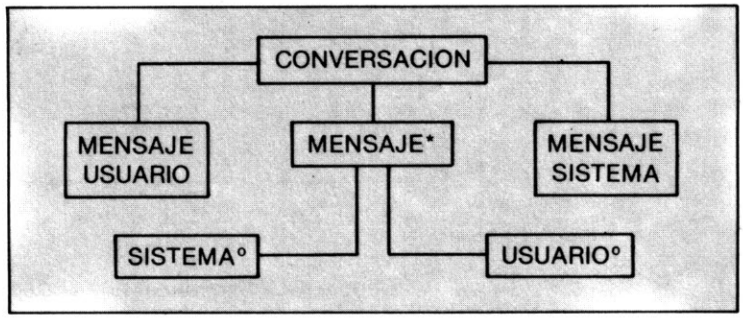
Una "conversación" entre el usuario y el sistema consiste en mensajes alternativos desde el usuario y desde el sistema. Supongamos, en este caso, que la "conversación" siempre comienza con un mensaje del usuario y siempre termina con un mensaje del sistema.

CONVERSACION

Mensaje del usuario
Mensaje*
sistema°] Selección
usuario°]
Mensaje del sistema

Iteracción

Secuencia



Hemos analizado las estructuras de los datos, aplicándolas a diversos casos de la realidad.

El buen manejo de estos conceptos es importante para el diseño de programas. Una de las claves para diseñar un buen programa es darle a éste una estructura equivalente a la de los datos que procesa.

Estos principios metodológicos pertenecen a Jackson, expuestos en su libro "Principles of program design".

Supongamos que tenemos que instruir al computador para que resuelva el siguiente problema en lenguaje TIMBA.

Dadas tres pilas no vacías de cartas —A, B y C—, ordenar los topes de menor a mayor. O sea que en el tope de pila A quede la carta más chica, y en el tope de pila C, la mayor.

Ej.

	A	B	C		A	B	C
Tope	9	2	6		2	6	9

En este caso. la estructura de los datos es clara, son tres pilas de cartas.

La forma elemental del algoritmo es:

Vo: (quiere decir: "versión cero")
Comienzo
Ordenar los topes de las pilas
A, B y C de menor a mayor.
Fin

Sabemos que no existe una sentencia TIMBA para realizar esto.

Para seguir descomponiendo, vamos a colocar la carta más chica en la pila A

V1

Comienzo

Buscar la carta más chica de las tres y colocarla en la pila A.
Ordenar los topes de las pilas B y C

Fin

Nuevamente no existe una sentencia TIMBA para realizar esto.

V2:

```
Comienzo
  Tome de pila A
  Si carta mayor valor que pila B
     intercambiar la carta con el tope de pila B
  Sino nada más
  Si carta mayor valor que carta de pila C
     intercambiar la carta con el tope de pila C
  Sino nada más
  Deposite en pila A
  Ordenar los topes de las pilas B y C
Fin
```

Hemos dejado la carta más chica en pila A (aunque aún no está totalmente escrito en **TIMBA**) bastaría con ordenar B y C.

V3:

```
Comienzo
  Tome de pila A
  Si carta mayor valor que pila B
     intercambiar la carta con el tope
     de pila B
  Sino nada más
  Si carta mayor valor que tope de pila C
     intercambiar la carta con el tope
     de pila C
  Sino nada más
  Deposite en pila A
  Tome de pila B
  Si carta mayor valor que tope de pila C
     intercambiar la carta con el tope de
     pila C
  Sino nada más
  Deposite en pila B
Fin
```

Ahora sigan descomponiendo jerárquicamente, hasta completar un programa en lenguaje TIMBA.

Ejercicios

Instruir al computador para que resuelva los siguientes problemas en lenguaje TIMBA. Desagreguen jerárquicamente hasta completar un programa en TIMBA (en todas las pilas mencionadas las cartas se definen boca arriba, a menos que el enunciado indique explícitamente otra cosa):

1. Dada una pila ORIGEN, pasarla a otra pila DESTINO, donde las cartas queden en orden inverso del dado.
2. Dada una pila ORIGEN, pasarla a otra pila DESTINO, donde las cartas queden en el mismo orden.
3. Dada una pila ORIGEN, reacomodarla para que las cartas queden en orden inverso.
4. Se da una pila POZO cuyas cartas pueden estar boca arriba o boca abajo, reacomodarlas para que queden todas boca abajo.
5. Dada una pila INICIAL, pasar la base al tope dejando el resto en el mismo orden.
6. Eliminar de una pila DADA todos los caballos.
7. Depurar una pila FUENTE eliminando los unos, los dos y los tres.
8. Dada una pila POZO eliminar oros y copas.
9. Seleccionar de una pila DADA las figuras de espadas.
10. Una pila MAZO contiene un mazo completo; formar otras cuatro pilas separando las cartas de MAZO por palos.
11. Eliminar de una pila dada el primer as (si hay alguno).
12. Eliminar de una pila dada el último as (si hay alguno).
13. De una pila que contiene un mazo completo separar los dos primeros oros que aparezcan.
14. De una pila que contiene cualquier cantidad de cartas separar los dos primeros oros, si hay suficientes, o los que se encuentren, en su defecto; el resto de la pila debe quedar en el mismo orden.
15. Seleccionar de una pila A la o las cartas que tengan el mayor valor.
16. Seleccionar de una pila A las ternas de cartas que contiguamente aparezcan sus valores en la siguiente relación: Anterior Actual = Siguiente.
17. Dada una pila A donde sólo existen cartas de oros y bastos, generar otra pila B donde aparezcan intercalados los oros y bastos (pudiendo sobrar oros o bastos).

Avanzando en el estudio de la Computación Uds. lograrán integrar mejor los conocimientos adquiridos.

Les proponemos que estudien el contenido de esta Unidad cuidadosamente, y luego lo analicen en grupo con la conducción del coordinador.

Cuando finalicen la tarea constaten si han accedido al logro de los siguientes

• Objetivos Generales

1. Conozcan el concepto "variable".
2. Comprendan ese concepto.
3. Conozcan distintas estructuras de datos.
4. Comprendan las distintas formas de agrupar las "variables", y que las estructuras logradas poseen características específicas.
5. Apliquen, con la técnica de descomposición jerárquica, los conocimientos adquiridos sobre estructuras de datos, para solucionar un problema.
6. Valoren Uds. la información adquirida.

• Objetivos Específicos

1.1. Definan el concepto "variable".
1.2. Definan el concepto "asignación".
1.3. Caractericen las "asignaciones".
1.4. Caractericen la "lectura".
2.1. Expliquen el concepto "variable", en función del trabajo del maestro computador.
3.1. Indiquen la denominación asignada a las formas de agrupación de las variables.
3.2. Describan las Filas.
3.3. Describan las Listas.
4.1. Comparen las características de las Pilas, Filas, y Listas.
5.1. Resuelvan un problema con la estructura de datos llamada Lista, descomponiendo jerárquicamente.
6.1. Participen activamente en el aprendizaje grupal, formulando sus dificultades.
6.2 Juzguen la importancia de la lectura cuidadosa previa del texto, para el éxito del aprendizaje en el grupo.
6.3 Juzguen la importancia de la autoevaluación.

Introducción

En la Unidad anterior (N° 5), al analizar la técnica de la Descomposición Jerárquica, aprendimos a transitar el proceso que va desde el algoritmo hasta las sentencias ejecutables en un lenguaje. Las sentencias ejecutables manipulan datos, cuyas estructuras determinan la estructura de los programas.

Hasta el momento sólo hemos visto la PILA como única estructura de datos. A continuación, veremos otras estructuras que nos permitirán avanzar en la solución de problemas.

Veremos también que la unidad elemental de los datos es la variable, a la cual podemos referirnos en el curso de los procesos.

1. Variables

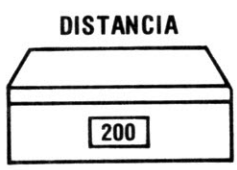
En TIMBA, hacemos programas que manipulan pilas de cartas, y que funcionan para cualquier carta contenida en las pilas.

Las pilas en TIMBA son un conjunto de cartas.

Las variables tienen asignado un valor único, que puede ser una letra, un nombre, un número, y al igual que las pilas de cartas las reconocemos por su nombre.

Una variable puede asumir distintos valores durante la ejecución de un programa.

Podemos pensar en una variable como un lugar dentro del computador que tiene **un nombre** y que puede almacenar **un valor**.

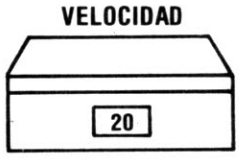


Es muy importante que conozcamos y recordemos la distinción entre el nombre asociado a una variable y su valor actual.

Veamos un ejemplo:

El nombre de la variable es PEPE, y el valor actual es 1.5. El nombre de la variable es permanente. El valor es algo más volátil y puede cambiar muchas veces durante la ejecución de un programa.

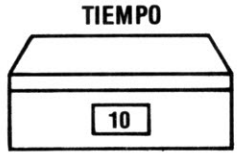
Supongamos que queremos hacer un programa para calcular la fórmula:



$$\text{DISTANCIA} = \text{VELOCIDAD} \times \text{TIEMPO}$$

Entonces, en un modelo muy elemental de computadora, la variable podría representarse con un casillero o caja, provista de una ventanilla para leer su valor en un momento dado. Al mismo tiempo, esta caja tendría una tapa que nos permitiría abrirla, cambiar el valor actual reemplazándolo por otro nuevo, y luego cerrarla.

Muchas de estas cajas, con ventanas para leer y tapas para cambiar valores, representarían la memoria del computador.



Imaginemos a una computadora como un cuarto con un gran número de cajas, donde trabajan el maestro computador y sus ayudantes: el asignador y el lector. El maestro computador simula la UCP y los ayudantes simulan los circuitos que interconectan la memoria y la UCP (recordar lo leído en la Unidad II).

Al escritorio del maestro computador llegan las instrucciones, y de acuerdo a ellas delega ciertas tareas a sus ayudantes.

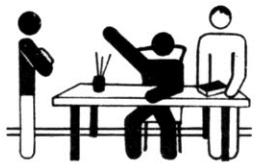
Supongamos que la computadora se encuentra en la mitad de la ejecución del algoritmo.

Una instrucción del algoritmo es:

$$\text{DISTANCIA} \longleftarrow \text{VELOCIDAD} \times \text{TIEMPO}$$

(Asignar a la variable llamada DISTANCIA el valor obtenido al multiplicar el valor de la variable llamada TIEMPO por el valor de la variable llamada VELOCIDAD).

Las variables VELOCIDAD y TIEMPO tienen asignados los valores que se muestran en el dibujo anterior de las cajas.



La operación requerida para la asignación aparece del lado derecho de la flecha, de manera que el maestro computador mira primero ese lado. Ve que debe conocer los valores de las variables VELOCIDAD y TIEMPO y ordena a su ayudante lector que le traiga esos valores de la memoria.

El lector se dirige a la memoria, identifica las cajas correspondientes a las variables, lee por sus ventanas los respectivos valores, los anota y entrega al maestro computador.

El maestro computador calcula el valor de VELOCIDAD × TIEMPO utilizando los valores de estas variables aportadas por el lector.

$$20 \times 10 = 200$$

A continuación, el maestro computador mira el lado izquierdo de la flecha en su instrucción.

DISTANCIA ⟵ VELOCIDAD × TIEMPO

Debe asignarle a la variable DISTANCIA el valor 200, que lo escribe en un papel y se lo entrega a su ayudante asignador, para que proceda.

El asignador va a la memoria, identifica la caja correspondiente a la variable DISTANCIA, abre su tapa, cambiando el valor que tenía, escribiendo adentro el valor calculado y cierra la tapa.

Vuelve junto al maestro computador para ejecutar una nueva tarea.

Vemos que la asignación es el proceso de asignarle un valor a una variable.

Decimos que **las asignaciones son destructivas**, porque destruyen el valor que la variable tenía hasta entonces.

La lectura, en cambio, es **no destructiva**, porque el proceso no altera de ninguna manera los valores de las variables de la memoria.

La variable es el elemento básico que se puede agrupar de distintas maneras. La PILA es una de ellas, y a continuación estudiaremos dos más, entre las muchas existentes.

Estas formas o modos de agrupación de las variables se llaman estructuras de datos, teniendo cada una sus propiedades y características.

Filas

Es la más sencilla de todas las estructuras. Mientras que en la pila los datos insertados más recientemente son los primeros en ser extraidos, en la FILA se respeta el "orden de llegada"

Esto hace que **la cabeza de fila** sea un dato privilegiado (el resto de la fila suele llamarse, por lo general, "cola"), y se define como el primero de los datos insertados que aún no ha sido extraído.

Las inserciones representan datos añadidos al final (cola) de la fila y las extracciones eliminan la cabeza de la fila.

En las pilas, en cambio, recordemos que tanto las inserciones como las extracciones de datos se hacen por el tope.

En ambas, fila y pila, estas operaciones de inserción y extracción son las únicas posibles.

Listas

Las pilas y filas tienen propiedades sencillas e interesantes, pero son de aplicación restringida.

Si quisiéramos saber, por ejemplo, cual es el segundo dato de una fila, sin perturbar el orden de la misma, tendríamos que recurrir a un complicado algoritmo que hiciera circular toda la fila para conservar el orden y poder determinar el valor de este segundo dato, al pasar por la cabeza.

Estas estructuras son muy limitadas para resolver problemas cotidianos y más complejos de información, lo que plantea la necesidad de disponer de estructuras más adecuadas y generales.

Son muchas las estructuras de datos existentes, aunque nosotros consideraremos unas pocas. Entre ellas está la llamada **LISTA**.

La lista tiene, como la fila, una cabeza y, como en el caso de pilas y filas, se puede extraer e insertar en ella un elemento.

La diferencia consiste en que la inserción y la extracción puede realizarse también en función del contenido de un elemento, independientemente de su ubicación dentro de la estructura. La lista no se altera si buscamos un elemento, ya sea por su ubicación, ya sea por su contenido.

Podemos:

— tomar el k-ésimo elemento de la lista
—examinar el contenido
—cambiar el contenido
—insertar en el k-ésimo lugar
—eliminar el k-ésimo elemento

Un ejemplo de lista puede ser un fichero. Cada ficha es una variable con sus contenidos. El conjunto de las fichas conforman una estructura lineal, con sus dos extremos. Sin embargo, podemos acceder directamente a una ficha, cualquiera sea su ubicación, extraerla, modificar o no su contenido y re-insertarla, o también eliminarla definitivamente.

Resumiendo:

Hasta ahora hemos visto tres estructuras lineales de datos. Según sea la forma en que los datos se insertan, extraen o eliminan, tenemos:

a) **La Pila** - Es una estructura lineal, donde las inserciones y extracciones se hacen por un extremo llamado TOPE.
b) **La Fila** - Es una estructura lineal, donde las inserciones se hacen por un extremo (cola) y las extracciones por otro (cabeza).
c) **La Lista** - Una estructura lineal que nos ofrece un margen mucho mayor de operaciones sobre los elementos que la componen, por cuanto podemos referirnos directamente a cualquiera de ellos.

3. Ejemplo

Proponemos a continuación el desarrollo de un ejercicio, donde aplicaremos la técnica de la descomposición jerárquica y los conceptos sobre estructuras de datos que hemos visto.

La tarea del programador consiste en instruir a una computadora para que imprima los primeros mil números primos, considerando al 2 como el primero.

Recordemos que, número primo, es un entero positivo que sólo es divisible por sí mismo y por la unidad.

La forma elemental del algoritmo es:

Versión 0:

Comienzo

"Imprimir los primeros mil números primos"

Fin

Si en un lenguaje de programación existiera una sentencia como la escrita en esta versión, habríamos resuelto el problema.

Pero supongamos que no existe

Nuestra propuesta inicial es separar la generación de los primos de su impresión, que se enuncia en la versión 0.

Entonces, la versión 1 quedaría así:

V_1
Comienzo
  Definir la lista de primos
  Llenar la lista de primos con los primeros mil números primos
  Imprimir la lista de primos
Fin

Naturalmente, si las órdenes que aquí damos están contenidas en el repertorio de sentencias del lenguaje que tenemos, entonces, el problema está resuelto.

Si no, continuamos.

El paso siguiente consistirá en determinar cuál será la estructura de la lista de números primos.

Será una lista de mil "casilleros", donde iremos guardando cada primo en orden ascendente. Usaremos una **estructura de lista** para poder referirnos a cualquiera de sus componentes de un modo directo.

En el primer casillero estará el 2, en el segundo el 3, en el tercero el 5, etc.

Valor de los números primos	2	3	5	7	11	13	
Nro. de orden de los números primos	1	2	3	4	5	6	... hasta mil

Cuando queramos referirnos al número primo 5 diremos que es el contenido del tercer casillero de la lista Primos, y anotaremos: PRIMOS (3)

O sea: PRIMOS (1) ⟵ 2, PRIMOS (2) ⟵ 3, PRIMOS (3) ⟵ 5
PRIMOS (4) ⟵ 7, PRIMOS (5) ⟵ 11, etc...

De manera general, PRIMOS (k) será el k-ésimo componente de la lista de primos. K será una variable que podrá contener cualquier valor entero entre 1 y 1000.

Haciendo una comparación, los casilleros serían como casas, todas ubicadas sobre la Avda. PRIMOS, y cada una con su número de dirección. El contenido de cada casillero (un determinado número primo) sería la familia que habita en una determinada casa.

Vayamos entonces a la versión dos.

V_2:
Comienzo
a1. Definir PRIMOS (k) con $1 \leq k \leq 1000$
b1. Para k de 1 a 1000, hacer primos (k) igual al k-ésimo número primo.
c1. Imprimir PRIMOS (k) para k de 1 a 1000
Fin

Ahora desarrollamos el punto b1 de la versión 2 (y lo designamos V_2b_2)

```
V2b2
Comienzo
Definir los enteros k, pospri; k ⟵ 0, pospri ⟵ 1.
Mientras k ≤ 1000
    Incremente pospri hasta el próximo número primo
    k ⟵ k + 1
    PRIMOS (k) ⟵ pospri
Trasmien
Fin
```

Pospri es otra variable cuyo nombre procede del "posible próximo primo", que estamos buscando.

Pospri es un nombre arbitrario, ideado por razones mnemotécnicas.

La iteración se realiza entre las dos palabras claves "mientras" y "trasmien" (semejante a paréntesis).

Pero vamos a suponer, nuevamente, que la sentencia "incremente pospri hasta el próximo número primo" no pertenece a ningún lenguaje.

Tendremos, entonces, que diferenciar el proceso de pasar al próximo entero y el de decidir si es o no primo. La variable pospri contendrá el valor del próximo entero, al que someteremos a un mecanismo para determinar si es o no primo.

El desarrollo de V_2b_2 lo designamos V_2b_3.

```
V2b3
Comienzo
  Definir los enteros k, pospri; k ⟵ 0; pospri ⟵ 1
  mientras k< = 1000
    - pospri ⟵ pospri + 1
    - decidir si pospri es primo, designando:
      resto = 0 si pospri no es primo y
      resto ≠ 0 si pospri es primo
    - si resto ≠ 0
        k ⟵ k + 1
        primos (k) ⟵ pospri
      sino nada más
  trasmien
Fin
```

En este punto podemos reflexionar varias cosas. Primero, que salvo el número 2, ningún par es primo, ya que todos son divisibles por 2. Por lo tanto, examinaremos sólo el 3, 5, 7, 9, 11, 13, 15, etc.

Segundo, que cada nuevo entero impar que debamos examinar basta dividirlo por primos menores que él (por la propiedad de descomposición en factores primos). Por lo tanto, tomaremos sólo los impares, y luego los dividiremos por los primos menores. En el caso de que todos esos cocientes tengan **resto** distinto que cero, diremos que es primo.

Resulta cómodo, entonces, diferenciar el 2 y el 3.

Así tomaremos el 11 y veremos:

$11 \div 3 = 3$; resto $= 2$
$11 \div 5 = 2$; resto $= 1$
$11 \div 7 = 1$; resto $= 4$

Por lo tanto 11 es primo.

Ahora tomemos, por ejemplo, el 49.

$49 \div 3 = 16$; resto $= 1$
$49 \div 5 = 9$; resto $= 4$
$49 \div 7 = 7$; resto $= 0$

Ya no tiene sentido ver qué pasa con los primos mayores que 7 y menores que 49, pues, al ser divisible por 7 ya no es primo.

Nota:

Cuando estamos dividiendo queremos ver cómo es el resto de la división entera, sin sacar decimales.

Recordemos que:

dividendo = (divisor × cociente) + resto

Por lo tanto:

resto = dividendo − (divisor × cociente)

```
V2b4
comienzo
  definir los enteros k, j, resto, cociente, pospri
  k ⟵ 2
  pospri ⟵ 3, PRIMOS (1) ⟵ 2, PRIMOS (2) ⟵ 3
mientras k ≤ 1000
  j ⟵ 1
  resto ⟵ 1
  pospri ⟵ pospri + 2
mientras resto ≠ 0 y j < k
  j ⟵ j + 1
  cociente ⟵ pospri / primo (j)
  resto ⟵ pospri − (primo (j) × cociente).
trasmien
  si resto = 0
  k ⟵ k + 1
  Primo (k) ⟵ pospri
sino nada más
  trasmien
F i n
```

Nota: a resto le asignamos artificialmente el valor 1, para que desde la primera vez entre siempre en el mientras.

Resumamos el significado de todas las variables utilizadas.

— **Primos:** es la lista donde se almacenarán los primeros mil números primos.

— **k:** es una variable que tomará los valores de 1 a 1000 y que servirá, cada vez, para indicar en qué **posición de lista** hay que almacenar el primo identificado a su turno.
Ejemplo: luego de determinar que el 11 es primo, k tendrá el valor 5, ya que es el quinto número primo.

— **j** es una variable que, para cada valor pospri, irá tomando sucesivamente el valor de las posiciones de los primos ya almacenados.

valor del número primo	2	3	5	7	11
número de orden del número primo	1	2	3	4	5

k = 5
pospri = 13
j = 2, dividimos 13 × 3, resto 0
j = 3, dividimos 13 × 5, resto 0
j = 4, dividimos 13 × 7, resto 0
j = 5, dividimos 13 × 11, resto 0
entonces pospri es primo

k = 6
primos (6) = 13
pospri = 15
j = 2, dividimos 15 × 3, resto = 0
pospri ya no es primo, k sigue valiendo 6
k = 6
pospri = 17
j = 2, dividimos 17 × 3, resto 0, etc.
j vuelve a ciclar

— **Resto.** Es una variable que tiene el valor del resto de la división del posible primo por los primos menores que él.

— **Cociente.** Es una variable que tiene el valor del cociente de la división del posible primo por los primos menores que él.

— **Pospri.** Es una variable que tiene el valor del número candidato a determinar si es primo.

Ejercicios

Estructuras de filas y listas.

1) Dada una fila de elementos numéricos:
   a) Imprimirlos cambiados de signo y en el mismo orden
   b) En orden inverso sin cambiar de signo

2) Dada una fila de elementos numéricos (distintos de cero), generar una fila de salida donde se encuentren los elementos cambiados de signo, un cero y los duplos de los mismos.

3) Redactar un pseudocódigo tal que si el primer elemento de una fila es un natural imprima el sucesor de todos los elementos de la fila, en caso contrario, imprima la cantidad de elementos de la fila.

4) Redactar el pseudocódigo de un programa que imprima el triplo de los datos leidos de una fila, a menos que se detecte un número negativo, en cuyo caso se suspende de inmediato la impresión.

5) Dada una fila con números enteros distintos de cero, separar en dos filas: POS y NEG, donde se encuentren los positivos y los negativos e imprimir los positivos seguidos de los negativos separados con un cero, y finalmente imprimir la suma total de los elementos.

6) Dados los elementos de una fila, generar una pila con los elementos mayores que 7 y luego imprimirlos.

7) Dada una fila de números naturales construir otra, donde aparezcan los pares y los impares separados por un cero e imprimirlos, imprimir además la cantidad de elementos pares y la de impares.

8) Dada una fila de números naturales imprimirla ordenada de menor a mayor si la suma de sus elementos es número par, u ordenada de mayor a menor si dicha suma resulta un número par.

9) Se tienen dos filas: Fila 1 y Fila 2, se pide obtener como salida la suma de los elementos de ambas filas si ambas filas tienen un número par de elementos, o la impresión de ambas filas con los elementos correspondientes en orden inverso al de entrada, en otro caso.

10) Dada una lista de 20 elementos:
   a) Sumar el contenido de los elementos pares (2, 4, 6, 8, 20)
   b) Sumar el contenido de los elementos impares
   c) Reemplazar el contenido de los elementos pares por el cuadrado de los mismos.
   d) Idem para impares

11) Dadas 2 listas de 10 elementos, generar otra cuyos elementos son la suma de los otros dos.

Ejemplo: A = (10, 20, 30, 1, − 1, 5, 6, 7, − 3, 0)
B = (0, − 8, 5, 4, 3, − 1, 0, 20, 100, 8)
A + B = (10, 12, 35, 5, 2, 4, 6, 27, 97, 8)

12) Buscar los elementos iguales a cero e indicar su posición

13) Hallar el menor de los números de una lista y el lugar que ocupa.